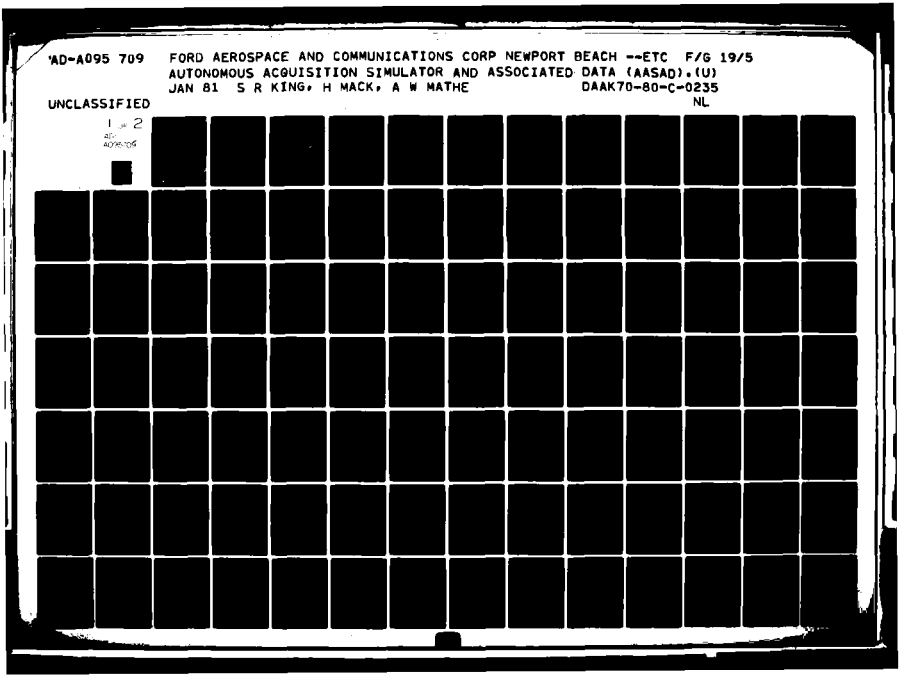
AD-A095 709 FORD AEROSPACE AND COMMUNICATIONS CORP NEWPORT BEACH --ETC F/G 19/5
AUTONOMOUS ACQUISITION SIMULATOR AND ASSOCIATED DATA (AASAD).(U)
JAN 81 S R KING, H MACK, A W MATHE DAAK70-80-C-0235

UNCLASSIFIED NL

1 of 2



AUTONOMOUS ACQUISITION SIMULATOR AND ASSOCIATED DATA

BY

S. R. KING
H. MACK
A. W. MATHE
A. S. POLITOPOULOS
J. R. BERCHTOLD
K. R. TAYLOR
J. N. CAST

FORD AEROSPACE & COMMUNICATIONS CORPORATION
AERONUTRONIC DIVISION
NEWPORT BEACH, CALIFORNIA 92663

JANUARY 15, 1981

First Quarterly Scientific & Technical Report
for Period 30 September 1980 - 31 December 1980

APPROVED FOR PUBLIC RELEASE
DISTRIBUTION UNLIMITED

Prepared for

U.S. ARMY MOBILITY EQUIPMENT
RESEARCH AND DEVELOPMENT COMMAND
NIGHT VISION AND ELECTRO-OPTICS LABORATORY
FORT BELVOIR, VIRGINIA 22060

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO. AD A095709	3. RECIPIENT'S CATALOG NUMBER --
4. TITLE (and Subtitle) AUTONOMOUS ACQUISITION SIMULATOR AND ASSOCIATED DATA (AASAD)		5. TYPE OF REPORT & PERIOD COVERED First Quarterly Scientific & Tech, 9/30/80-12/31/80
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) S. R. King, A. S. Politopoulos, J. N. Cast, H. Mack, J. R. Berchtold, A. W. Mathe, K. R. Taylor		8. CONTRACT OR GRANT NUMBER(s) DAAK70-80-C-0235
9. PERFORMING ORGANIZATION NAME AND ADDRESS Ford Aerospace & Communications Corporation Aeronutronic Division Newport Beach, California 92663		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS U.S. Army Mobility Equipment Research & Development Command Night Vision and Electro-Optics Laboratory Fort Belvoir, Virginia 22060		12. REPORT DATE January 10, 1980
		13. NUMBER OF PAGES 168
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

digital video preprocessors
intermediate high level language
macrocode
microcode
FLIR
target classification
video signal processor
target cueing
target screening
target acquisition

(Continued)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report is the first quarterly scientific and technical report for the Autonomous Acquisition Simulator And Associated Data. Contained herein is an introduction or executive overview of the AASAD system and an architecture review of two systems studied and considered in the architectural design of AASAD. This report also defines the programming structure and its relationship with the hardware modules of the system. The Digital Video Preprocessors are described and the planned activity for the contract second quarter period is outlined.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

19. Key Words (Continued)

target tracking
target detection
VLSI
real time processor
image processing
image enhancement
target handoff
target extraction

CONTENTS

ILLUSTRATIONS

TABLES

SECTION 1

INTRODUCTION AND SUMMARY

INTRODUCTION

This is the first Quarterly Scientific and Technical Report for the Autonomous Acquisition Simulator and Associated Data (AASAD), contract number DAAK70-80-C-0235. This report covers the activity period of 30 September 1980 to 31 December 1980. The following introduction provides an overview of the purpose and functionality of the simulator.

Imaging processing is an extremely difficult form of signal processing. The operational environment varies greatly in background, temperature and target physical characteristics, making target acquisition, tracking and fire control a difficult task. Depending on the sensor utilized the number of operations required in an image processing weapon system can vary anywhere from one million to 800 million per second. Today's computers and even special purpose processors were not designed to efficiently handle such extremes. A special purpose processing system that is modular in hardware and software can efficiently solve the image processing problems using existing algorithms and provide the development system for new algorithms.

The benefits realized from an effective image processing system are many. A dual mode missile combining radar for acquisition and image processing for tracking and aimpointing is much more effective against moving or stationary targets than a simple radar missile. In a high energy laser weapon system the fine tracking and pointing accuracy gained from image processing is necessary for the weapon to be effective. In an autonomous weapon system as required on today's fighter aircraft the automatic acquisition, classification and fire control functions place extreme loading and throughput requirements on today's signal processors. A modularly expandable signal processor like AASAD meets the needs of all three of these image processing weapon systems.

The AASAD hardware and software has been designed to meet the needs of image processing systems today and be versatile enough to ensure anti-obsolescence. Modular hardware/software architectural approach has been used in other forms of signal processing but AASAD is the first completely programmable modular image processor. The flexibility of the system allows it to efficiently and in near real time, execute a wide range of image processing algorithms, and be operationally changed for a completely different function simply by loading the new program.

The Autonomous Acquisition Simulator is a laboratory instrument designed for image processing of video data (TV or FLIR) in the standard 525/875 lines per frame format. The design objective of the simulator is to provide a system using state-of-the-art hardware, operating at sufficiently high speed and architecturally structured to achieve near real time operation with the video data. Configuration of the system, resource allocation and the algorithms performed are under program control and hence, the simulator can be completely reprogrammed for vastly different algorithms in a matter of seconds with no hardware changes required. The Autonomous Acquisition Simulator can be programmed such that the classical "man-in-the-loop" is not required for proper operational performance.

The Autonomous Acquisition Simulator can be used for performance testing of image processing algorithms either in the laboratory or in some field testing environments. The simulator is packaged to operate in the environment experienced in the OH-6, Model 500P helicopter and as such requires a primary power source of DC voltage.

The system has been designed to maximize the ease of programming, capabilities expansion and operation. Both the hardware and software are designed and developed on a modular basis. Some of the simulators salient features are:

- Unique modular Digital Video Preprocessors that implement dedicated software controllable image processing functions in real time and may be easily expanded to include additional dedicated preprocessors.
- Unique modular high speed digital microprocessor that uses pipelining and parallel processing for expansion from speeds of fifteen to several hundred million operations per second (MOPS) to meet real time processing requirements.
- Flexibility of programmable or software controllable processors distributed throughout the simulator.
- Pascal based Intermediate High Level Language for the generation of new application programs.
- Modularly expandable system architecture to meet real time processing requirements for future applications.
- Open ended Macro and Micro library.
- Serial and parallel data input and output.
- Input data sample rate up to 25MHZ.

The AASAD system is packaged in a twenty-five card-slot chassis. The basic configuration uses nineteen cards. The remaining six slots have four

dedicated to specific options (additional arithmetic processor (1), Control Processor memory (1) and Digital Video Preprocessors (2)) and two are available for user selected options.

The system provides the following interface capabilities:

- One video input and two video outputs.
- Two RS232 serial communications interfaces and one MIL-STD-1553A interface.
- One digital parallel interface with 16 address, 16 data and 3 control lines.
- One discrete flags interface with eight input and eight output lines.

The major functional elements of the Autonomous Acquisition Simulator and Associated Data (AASAD) system are shown diagramatically in Figure 1. The general characteristics of each functional area are:

(1) Program Interface. The program interface contains two distinct functional modules. The first module is the Program Loader Module which interfaces the host computer to the second module, the Program Transport Module. The Program Transport Module contains non-volatile memory in which the programs, to be executed in the AASAD system, are temporarily stored. The AASAD system can be reprogrammed offsite via the Program Transport Module.

(2) System Controller. This function is performed by the LSI 11/23 microcomputer. The System Controller provides the basic operating system, utilities, executive, intermediate high level language, and the application programs. The System Controller down loads the functional tasks to all the microprocessors and program controlled function modules in the AASAD system through the Control Processor. The System Controller also performs the basic math functions that require floating point operations.

(3) Control Processor. The Control Processor's operational software links the Micro library to the System Controller generated Macros for execution by the functional elements of the AASAD system. The Control Processor provides direct control over the programmable elements of the AASAD system and does so at the field rate. It allocates the resources required by the operation or algorithm specified by the System Controller. This processor/controller, like the others in the AASAD system, is designed as a high speed microprocessor.

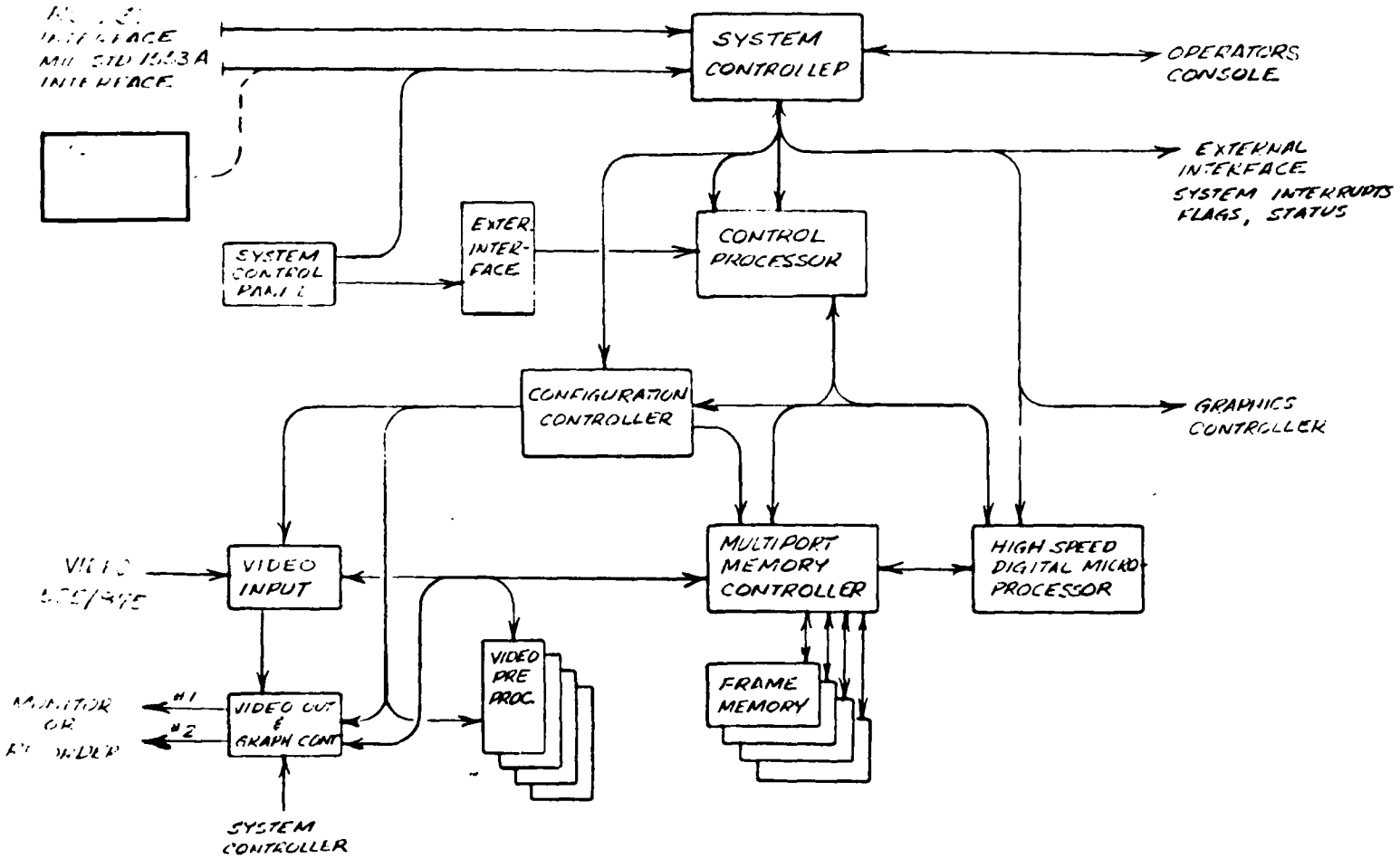


Figure 1. Simplified block diagram autonomous acquisition simulator

(4) Memory Subsystem. The memory subsystem consists of two multiport controllers and four frame memory cards. All access to and from the frame memories is through the high speed port controller and the low speed port controller. The frame memory cards are equipped with 256K bytes each and have expansion options to increase their capacity to 500K bytes or one megabyte.

(5) High Speed Digital Microprocessor. This subsection performs the high speed arithmetic operations required by various image processing algorithms. Each microprocessor is designed to use one to four parallel arithmetic elements to provide operational speeds up to 60 million operations per second (MOPS). One arithmetic element is supplied with the AASAD system and one card slot is dedicated for optional arithmetic elements.

(6) Video Input. This subsection accepts the analog video signal and separates the video data from the synchronization signals. The source of this data may be camera, sensor or magnetic tape. The video data is converted to digital values at data (pixel) rates up to 25MHZ. The data format accommodated is standard 525 or 875 lines per frame standard TV video.

(7) Video Output/Graphics Controller. The video output section can supply both 525 or 875 lines per frame analog video to one or two monitors. The Graphics Controller generates both symbols and text characters (full ASCII code set) at image locations specified by the Configuration Controller and overlays the symbols/text on the displayed image.

(8) Video Preprocessors. The preprocessors are program controlled by the Configuration Controller and provide the "on-line" or near real time operations on the data supplied by the video input or from frame memory or both. The Video Preprocessors are designed with very high speed ECL logic elements to facilitate the 25 MHZ data rate of the Video Input section. Four preprocessor functions are provided with the AASAD system; Variable Averaging Mask, Video Edge Detection Mask, Sum/Difference and Threshold/Binarization.

(9) System Control Panel. This function allows the operator to interact with some system operations such as symbol generation and placement, alteration of certain memory locations in the controller/processor elements, etc. The control panel is also designed with a micropro-

cessor which enhances its flexibility and synergistic capabilities with the AASAD system. The control panel will be used in the initialization and testing phases of operation and the operational mode of some algorithms.

For a more detailed description of the operational capabilities of these functional modules/subsections please refer to the appendix of this report.

SUMMARY

This report contains five sections and one appendix and, with the exception of the appendix, establishes the format for the following quarterly reports. Since this is the first report and covers the program initialization phase the technical accomplishments are not advanced to the level that provides significant scientific and technical details. Section 2 Algorithms, describes the "system trades" considered in the definition of the AASAD architecture. Specifically this study identified the Digital Video Preprocessor function which allows near real time execution of certain algorithms frequently used in image processing. The distributed microprocessor design to provide program control at each functional level was selected as a result of the study when other systems were evaluated as "difficult to alter their mission objectives."

Section 3 Software, defines the programming architecture and relates each program with the specific hardware that executes the program. Programming development uses "Top Down Structure" and utilizes a Pascal version for the Intermediate High Level Language (IHLL). The algorithm programs utilize an open ended library of macros and micros to formulate the desired operations. Section 4 Hardware, details the functional design and operational characteristics for the Digital Video Preprocessors:

- Sum/Difference and Threshold/Binarization
- Variable Averaging Mask
- Video Edge Detection Mask

These modules are presently undergoing system integration testing in the FACC Modular Video Image Processing System (MVIPS).

Also an analysis of the circuit card selected for use in the AASAD system is presented. The use of Multiwire[1] provides AASAD with the electrical

---

(1) Multiwire® is a registered tradename for Kollmorgen Corporation's discrete wired circuit boards.

properties necessary for the high speed ECL logic and for the weight, size and vibrational operation of the AASAD system in helicopters of the OH-6, Model 500P type.

Section 5 provides a "snapshot" of the activities that will be performed during the next report period, 1 January 1981 to 31 March 1981.

SECTION 2

ALGORITHMS

INTRODUCTION AND SUMMARY

This summarizes the results of system architectural trades that were made of two existing auto-cueing systems prior to finalizing the design of the AÂSAD system. The Honeywell Prototype Automatic Target Screener (PATS) and the Westinghouse Auto-Q Digital Processor are the two systems that were selected for this trade study. Some other architectures were considered and studied during this phase of the program, but the results are not reported here.

The approach to performing the trade study was the review of each system in terms of its ability to meet the requirements for implementing some of the existing algorithms for autonomous acquisition, i.e. computational speed, memory, etc. Consideration was also given to the flexibility or ease with which algorithms could be modified or new algorithms implemented on the systems.

The results of this study are reported in the following subsection by first reviewing the basic computational requirements for some well known autonomous acquisition algorithms. From these requirements, evaluation criteria were developed as a basis of comparison. The two selected systems along with the architecture of the AASAD system are then described in the context of these criteria. The results are summarized in tabular form at the end of this section.

SYSTEM ARCHITECTURE REVIEW

The Section examines the architectures of several image processing systems currently in existence for target detection and Auto-Cueing Applications. Specifically, it will concentrate on the real time aspects of implementing image processing algorithms and their implications on hardware and software systems. The three system architectures being examined here are the Westinghouse AUTO-Q system, the Honeywell PATS system and the Ford Aerospace MVIPS/ AASAD system.

Honeywell Approach (PATS)

(Reference: SPIE Proceedings "SMART Sensors" April 1979 page 125)

The Prototype Automatic Target Screener System (PATS) was developed by the Honeywell Corp. to perform automatic real time detection, classification, recognition, and cueing of tactical targets. Both analog and digital hardware are used to perform these major functions.

The hardware consists of twenty-two 6" x 9" boards and fits into a box slightly larger than an ATR box and has a power consumption of approximately 200 watts. Charge-coupled devices are used for performing image segmentation and a bit-slice microprogrammable digital processor (AMD 2903) is used to implement a target classification algorithm. The PATS system accepts 525 and 875 line TV formats and performs target classification and detection at a rate of 10 frames/second. The results of these tasks are conveyed to the operator by means of symbology overlaid on the TV display.

The functions performed by the PATS system on the incoming video data are shown in Figure 2. The image enhancement function consists of adaptive contrast-enhancement, DC restoration and automatic global gain and bias control and is implemented in analog hardware using CCD devices. Adaptive contrast enhancement circuitry enhances local variations of contrast and compresses the overall scene dynamic range to match that of the display. DC restoration eliminates streaking due to loss of correlation between lines because of the AC coupling of the detector channels. The automatic global gain and bias controls provide feedback signals to the FLIR device to maintain signal within the range of the E.O. multiplexer.

Target detection and classification consists of functions to detect targets and classify them into one of the five classes: 1) 2-1/2 ton truck, 2) tank, 3) APC, 4) track mounted radar-controlled anti-aircraft and 5) track mounted anti-aircraft missile launcher. These functions are realized using both analog and digital circuitry.

The hardware of PATS is shown in Figure 3. The first module performs sync stripping and video switching. From the composite sync signal derived from the incoming video, basic sync signals such as vertical reset, field indicator and horizontal sync are derived. The module switches the video to the monitor which can be either raw video, enhanced video, analog test signals or digital test signals. Also, the video to the rest of the system can be either raw or enhanced video. The module also contains circuitry to generate two clock signals -- a 455 clock for CCD devices and a 512 clock for the digital hardware -- synchronized with the incoming video. With 875 line format, there will be 512 or 455 samples per 32 μseconds whereas with 525 line format, the same number of samples will be obtained every 53 μseconds.

The image enhancement module performs DC restoration, global gain and bias control, and adaptive contrast enhancement using CCDs, a microprocessor, and MSI/LSI standard components.

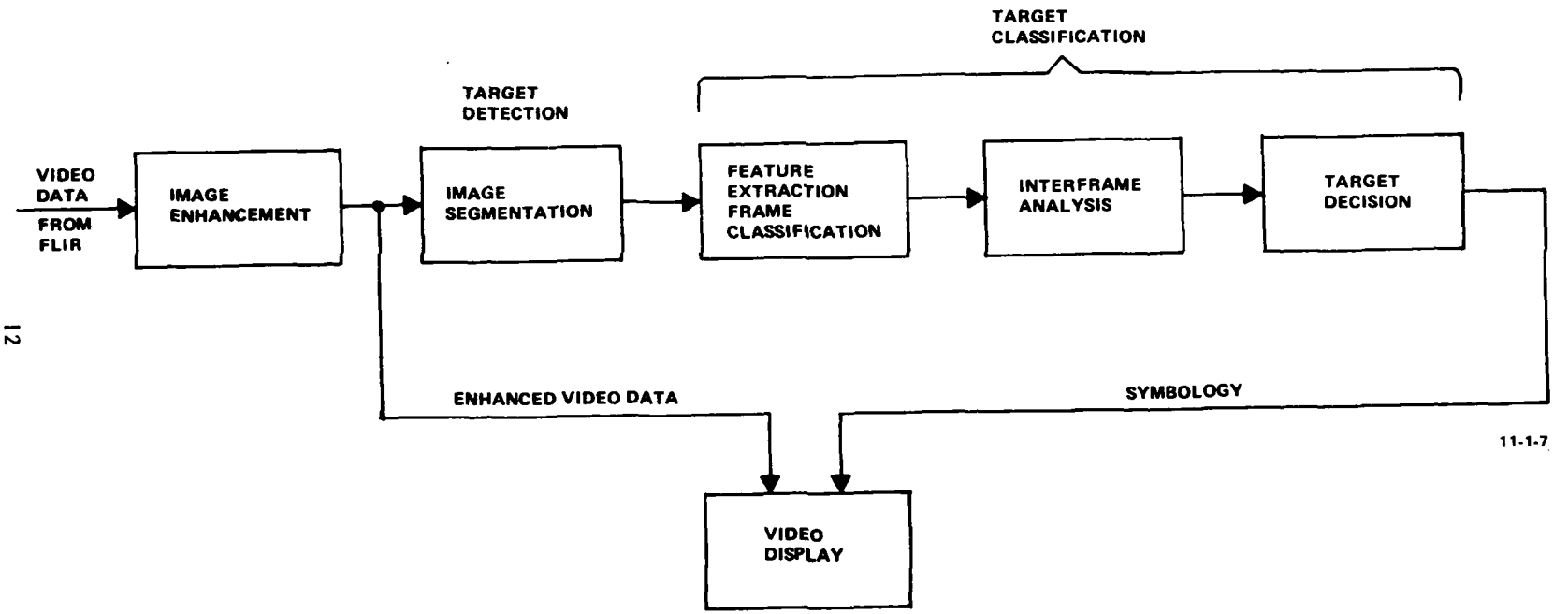


Figure 2. PATS functional diagram

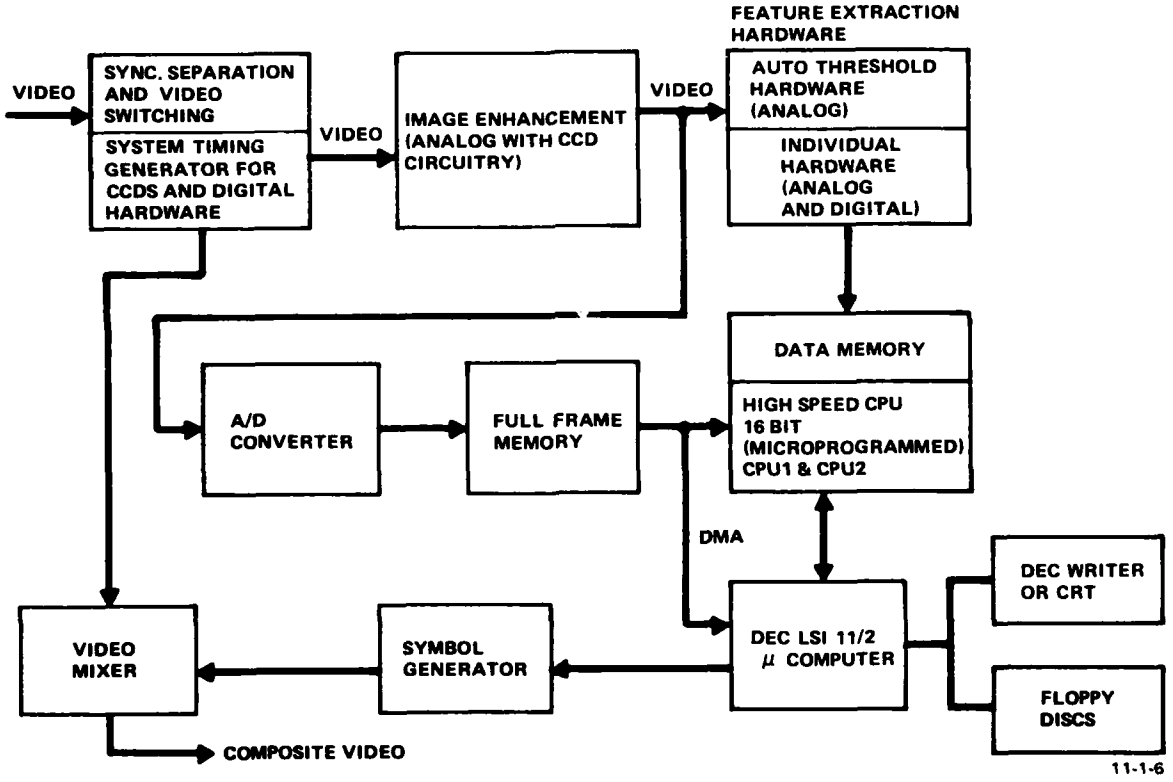


Figure 3. PATS hardware

The feature extraction hardware consists of two hardware limits, one for autothresholding and the other for generating intervals on a line-by-line basis which contain profiles of potential targets. The autothresholding unit generates two thresholds, one based on edge information and the other on background intensity. Based on these two thresholds, the unit generates "hot" or "cold" signals when the intensity exceeds or does not exceed background, and true or false signals indicating presence or absence of an edge at a given location. The thresholds are computed based on the previous and current lines. For the background threshold, a two-dimensional low-pass recursive filter is used to estimate the background average. After subtracting the background estimate from the video, the threshold is computed using the variance of the resulting video and a predetermined multiplication factor. The edge threshold computation is similar except that an edge filter is used to estimate the average edge intensities. The edge signal is computed using the following equation:

$$E(n,R) = I(n+1, k) + I(n, k) - 2I(n-1, k) - I(n+1, k-1) + 2I(n, R-1) - I(n-1, k-1)$$

where n and k are the scan line number and pixel position, respectively.

The digital signals (edge, hot/cold) from the autothresholding unit are passed to the interval hardware unit which generates valid intervals based upon previous, present, and next scan line edge/no edge, hot/cold signals. The interval unit also produces several first-level feature data stored in batches such as the line number, number of intervals for the line, starting position and width of intervals, background and intensity information associated with each interval, edge coincidence information for each interval, and indication whether the interval is "hot" or "cold".

The data from the interval generator unit is passed to the data memory which is part of the subsystem CPU1. The data memory can hold up to 2500 sets of interval data and a maximum of 21 intervals per line. The CPU1, built with AMD 2900 components, has a high speed 16 x 16 bit multiplier and is interfaced with a DEC LSI 11/2 and various memories (Figure 3). The DEC LSI 11/2 system is not used during the actual operating mode and is primarily used for training the PATS system.

Both CPU1 and CPU2 are interfaced to the symbol generation hardware which generates symbols with programmable size and shape. The A/D converter shown in Figure 3 contains two TRW A/D 8-bit converter chips which are used to determine the digital value of the background estimate at the beginning of each

interval, and digitize the entire video frame for storage in the full frame memory. A 12-bit adder is provided in the converter block which performs summation of intensities over the entire interval. The eight most significant bits are transferred to the CPU1 for processing. The full frame memory consists of eight 16K x 1 dynamic RAMS with access time of 375 ns and can be randomly accessed by CPU1 for calculations necessary for target recognition and classification.

The CPU1 performs several functions through software on the interval data generated by the interval unit. It combines the one-dimensional intervals into sets of connected intervals characterizing two-dimensional objects. It performs a one-dimensional median filter of width five on a line-by-line basis to smooth intervals and computes object features in an hierarchial manner. In the hierarchical process of feature computation, it eliminates clutter in initial stages and computes moment features for potential targets. The objects are classified using a k-nearest neighbor rule and the moment features into one of five categories. The processor CPU1 stores the result of the classification along with the object size and location so that this data can be further processed by CPU2 for interframe analysis. The interframe analysis consists of consistent object association between frames and application of Bayes' rule for object classification. CUP2 also performs symbol generation.

To conclude the architectural description of the PATS system, we note the ingenious manner of analog and digital division as well as hardware and software division to perform a given set of image processing tasks in real time. However, the main limitation is that it performs a restricted type of auto-detection and requires a total redesign of hardware and software to perform a different type of image processing algorithms (such as a double gate size contrast filter or a Sobel edge filter).

Also, the processing speed of the system is degraded as more targets are detected and cued by the system. Another possible problem or inconvenience with this system is the difficulty of obtaining intermediate results from the interior elements when the unit is processing. Having this feature would greatly facilitate debugging and checkout of system malfunctions.

Auto-Q Digital Image Processor (Westinghouse Corp.)

(Reference: 1980 NAECON Proceedings, Vol 1, pages 102-107)

The Auto-Q digital image processor was developed by Westinghouse to extract key image features from FLIR video data for target classification or for image registration (scene matching) in hand-off applications.

Details of the Auto-Q system are shown in Figure 4. It accepts data in either 525 or 875 line formats and processes at a rate of six fields per second for 525 line formats and three fields per second for 875 line formats.

The Preprocessor (Figure 4) is under software control and its configuration can be changed frame to frame. It can perform averaging, median filtering, level slicing (at two levels) and rate compression by selection of alternate pixels. After these preliminary pixel operations, a Roberts Cross Operator is performed on two-by-two pixel arrays to obtain gradient information. Pixels with gradient amplitudes greater than an adaptively selected threshold are marked and further processed by the maximize function.

The maximize function thins the gradient data into thin lines of the target by comparing neighboring gradient values for all pixels. The blob tracker attempts to find a closed boundary by tracking left and right outlines. The blob is detected when left and right outlines meet, and the geometric features of the blob are computed and stored during the blob tracking process. The segment tracker connects pixels with compatible features defining the end points and gradient direction of image edges.

Group formation is handled by dedicated hardware which collects all edges associated with each target or object. It accomplishes this function by looking at the identifying tags attached to both blobs and edges as they are generated. After the group formation, the data is passed to a general purpose computer (AN/AYK-15A) which extracts targets and performs various other functions as indicated in Figure 4. These functions include frame-to-frame integration, feature computation, statistical decision making, feature matching for handoff, target prioritization and target classification.

The Auto-Q prototype model fits in a standard ATR chassis and has an AC power consumption of 450 watts.

In contrast with the Honeywell PATS system, Westinghouse uses a general purpose computer to implement a majority of the image processing tasks required for target classification and handoff. Although this increases the system flexibility, it may cause a reduction in real time processing rate.

Ford MVIPS/AASAD Approach

The basic goal of the Ford MVIPS/AASAD system is to develop a very high speed, digitally orientated, software programmable, real time image processing system that provides the capability to execute, in real or near real time, a wide range of complex image processing algorithms used for autonomous tactical target acquisition applications.

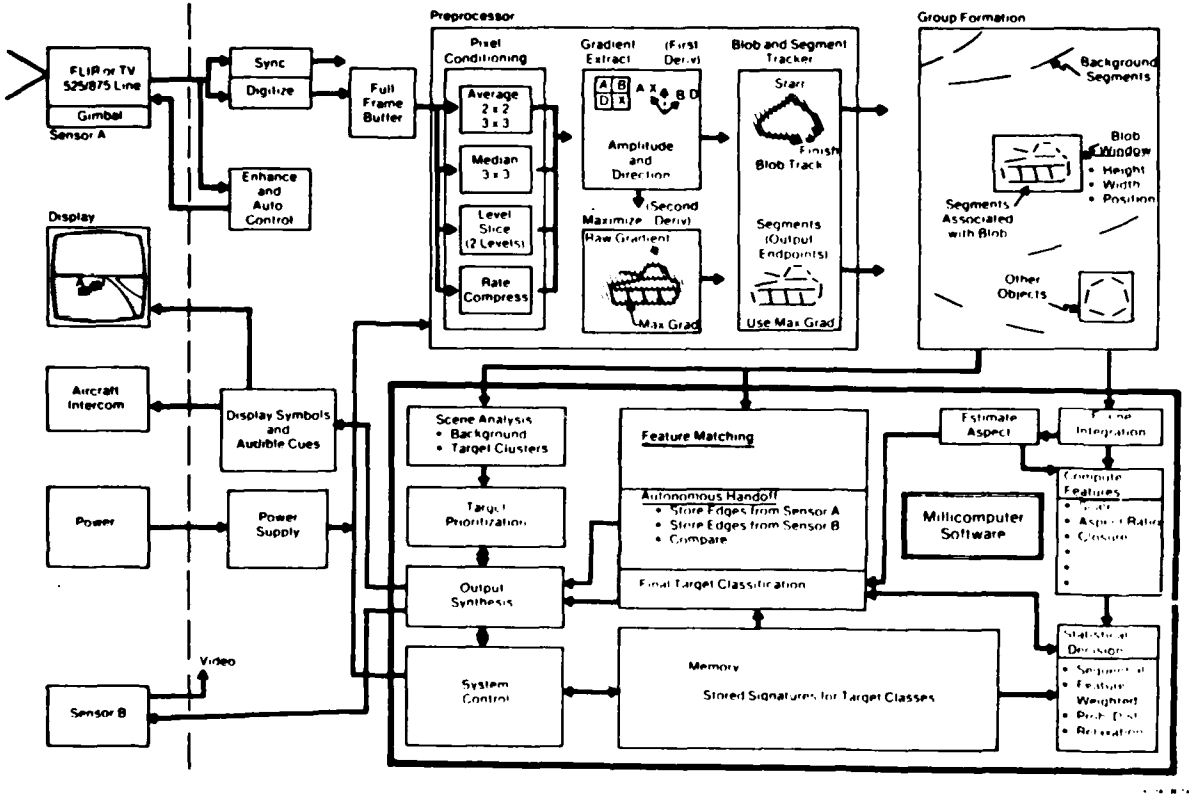


Figure 4. Westinghouse auto-Q digital image processor

The basic architecture chosen for this system, which is shown in Figure 5, contains a series of parallel pipelined digital processing units as the central processing elements. These elements are designed in a modular form such that additional units can be added to meet the more demanding requirements of some of the complex image processing algorithms it will be required to process.

One of the basic concepts of Ford's MVIPS/AASAD system is that the majority of the hardware units are software programmable from a high level, user orientated, language. This gives the system the computational power of other systems along with the added feature of software flexibility for ease of algorithm testing and validation.

The following section discusses some of the important features of the MVIP/AASAD system architecture.

Video I/O Preprocessor

The MVIP/AASAD system will accept raw input video in either 525 or 825 line TV format. This input video is then fed to a digital preprocessor which has a sync stripper, line drop computation and a D/A converter capable of converting video at a 25 MHz rate, at 8-bits per sample.

The video preprocessor has several additional features (built into the processor) that are software selectable and will perform real time image processing functions.

The unit can perform real time video field summation and subtraction, it can also perform a real time edge detection by executing various selectable operator masks such as Laplacian, Roberts and Sobel. A Variable Averaging Mask feature is available that will output the average pixel value from a selectable size window processed over an entire video field in real time. Finally, a threshold and binarization function is provided that takes preselected threshold values and outputs video pixel values of either 1, 0, or the actual pixel value based on threshold settings.

High Speed Digital Video Processing

The Ford MVIPS/AASAD system contains a High Speed Digital microprocessor (HSDM) device that is designed to perform the more complex algorithms that are required for autonomous acquisition and classification scenarios.

This unit is designed to operate independent of the video preprocessor and will process data in a parallel pipelined mode to provide a very high data throughput which is usually required for complex algorithm processing.

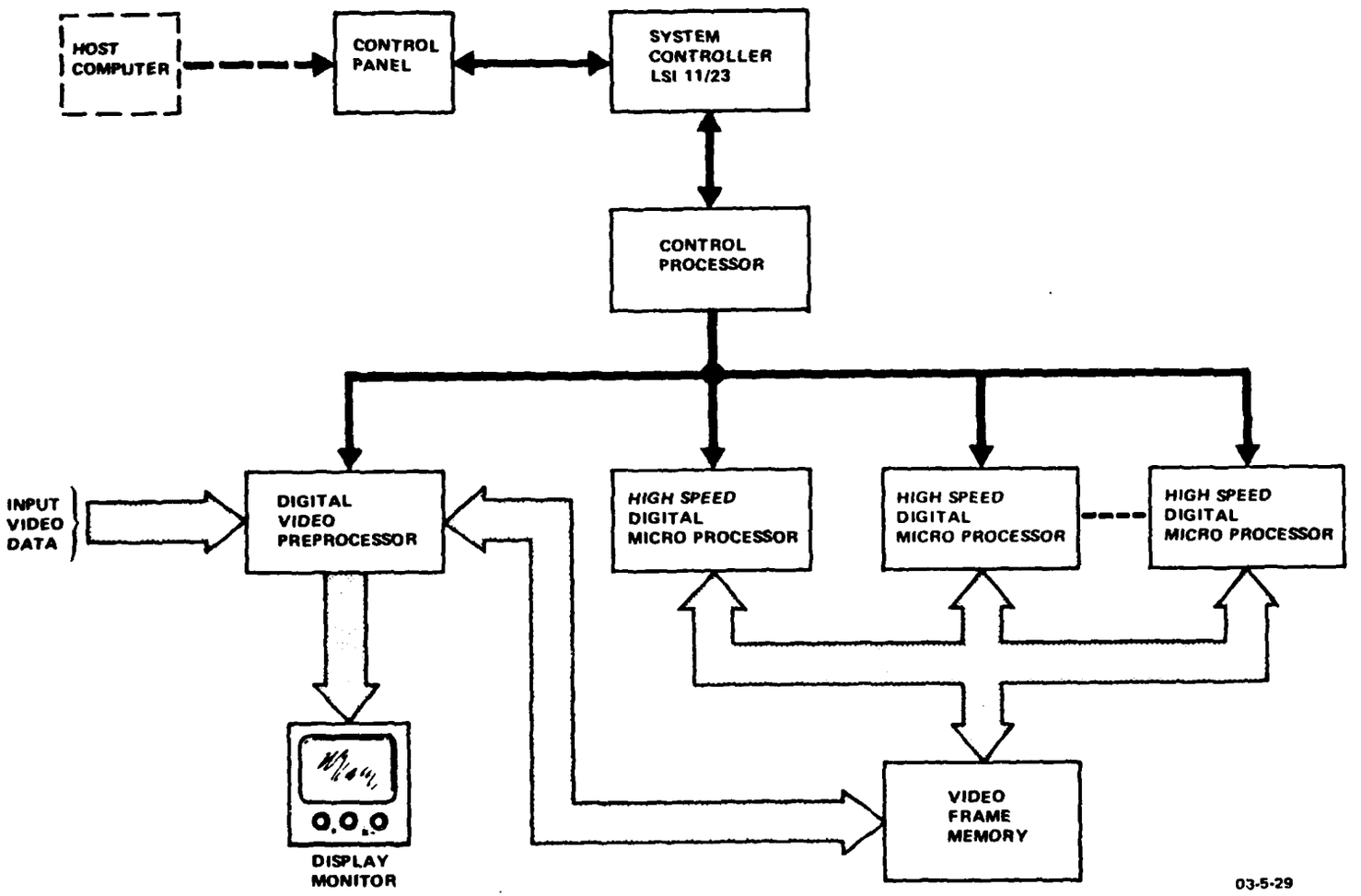


Figure 5. Basic system architecture

This unit is also software programmable and can perform such algorithmic functions as:

- 2-D recursive and nonrecursive filters
- Matched filters
- Feature matching
- Super slice technique
- Target segmentation
- Target classification
    - Statistical
    - Syntactical
- Multi mode tracking
    - Centroid
    - Correlation
    - Feature
    - Others

at real or near real time rates.

The HSDM unit contains a bit sliced device called the Pipelined Arithmetic Processor and is capable of processing video data at a rate of approximately 15 MOPS. It is interfaced to a video frame memory which stores input data from the video preprocessor and has a maximum capacity of 1 Mbytes per frame with four frames supplied.

The PAP units are controlled by the Control Processor unit which handles the general program flow and communicates with the PAPs at the micro instruction level.

The overall system data flow is handled by the System Controller. This device is actually a general purpose digital computer (LSI 11/23) and contains the system operating software. The unit is the basic interface to the user thru the control panel and the host computer or program loader. The following items are a few of the functions this unit will perform:

- Overall System Control
- High level language user interface
- Provide floating point computation
- Operating System Software
    - Compilers
    - Assembliers
    - Utilities
- Provide external signals for servo loop control

Ford MVIPS/AASAD Processing Concept

Since the Ford MVIPS/AASAD Architecture is almost entirely software programmable, the type of processing that is performed depends primarily on the nature and type of algorithm being implemented.

Thus, Honeywell's detection algorithm 'A' can be coupled with a Westinghouse classification algorithm 'B', along with a new tracking algorithm from source 'C' can be combined to determine if that configuration of algorithms is a suitable solution to a particular auto-cueing problem.

The basic concept of the Ford architecture is flexibility without a sacrifice in processing time and accuracy. User orientated software is also another keystone of this system.

Having a set of high level macro language calls (PASCAL) that are converted into micro instructions by the software complier and are executed as micro code in the HSDM, provides the user with the capability to program and examine the performance of a wide range of algorithms with a minimum amount of effort.

The development of the MVIPS/AASAD system used the following basic requirements for the design objectives of the hardware architecture and the structure of the operational software.

I/O Requirements

MVIPS must be capable of accepting analog data in TV format with provisions for eventual upgrading toward acceptance of other image data format. In addition, it should be capable of accepting both digital parallel data and digital serial communications data.

Digitized input data, intermediate results and final processed results should be capable of transfer to TV monitors, other computers (such as Navigation or Mission computers and Digital Servo Controller) and status indicators.

Preprocessing Requirements

MVIPS must be capable of executing a variety of preprocessing functions on the incoming imagery data. These include, image enhancement algorithms such as local area gain brightness control, and edge detection, noise smoothing through averaging, median and general recursive and nonrecursive filtering and sensor dependent processing aimed at correction of sensor related deficiencies such as auto-focus control, scan jitter elimination and detector array compensation.

The preprocessing functions are to be implemented for application over a complete frame of data. For man-in-the-Loop systems the preprocessing functions perform the necessary visual display improvements to enhance operator

effectiveness. For autoscreening/cueing systems the preprocessing functions are algorithm dependent but they also include the sensor associated processing.

Detection-Extraction Requirements

Given a frame of imagery MVIPS must be capable of scanning the scene on a pixel by pixel basis with any one of a variety of detection masks whose purpose is to isolate in the Field Of View (FOV) "target-like" areas or "areas-of-interest" that indicate the possible presence of a target object. The masks are under continuous evolution by different industrial and academic research organizations and thus MVIPS must not restrict the design to a specific approach. Instead it must allow sufficient flexibility to accommodate the existing approaches with additional consideration given to inevitable modifications.

Detection masks can assume a variety of forms implying both arithmetic and logical operations on the underlying image data covered by the mask. Operations such as thresholding, edge calculations, statistical parameter evaluations (mean, standard deviation etc.) histograms and logical interpixel comparisons must be within the MVIPS design framework. In addition. it must be possible to implement these different operations on different subsets of the pixels encompassed by the mask.

The fundamental goal of the detection phase is the localization of "areas-of-interest" which have sufficient characteristics to be potentially target objects. This ultimate characterization will be accomplished by the more sophisticated classifier algorithms. Hence, the detection phase must be viewed as a preprocessing phase which is designed to reject as much "clutter" as possible while retaining within its responses a subset of the scene elements containing possible target objects. In this sense, the detection algorithm designers strive to imbed within the detection mask characteristics, sufficient mathematical structure to capture the essential target discriminating properties but with a view toward keeping the arithmetic details to a minimum so that realistic execution times become feasible.

The detection masks can be of variable size to accommodate range dependent target size variations. In the limit, the whole scene may become the mask size. MVIPS must be able to adapt to these changes. That is, it must be possible to change the mask definition on a pixel by pixel basis during a scan across the FOV.

Proper performance of classifier algorithms require a clear definition of the localized object's geometric profile. Hence, for detection schemes which

do not clearly delineate the possible target object, there is a need for a second operational phase whose purpose is to extract the detected object from its surrounding background. MVIPS must be capable of accommodating such an extraction capability through operations, on a scene pixel subset, that include histograms, edge operations, statistical calculations, thresholding at one or more levels and logical comparisons.

Classification Requirements

Given a localized object and extracted from its immediately surrounding background object, MVIPS must be capable of implementing a host of classification algorithms whose purpose is the identification of the object under test. These are the algorithms that will finally ascertain the identity of the test object, i.e., is it a tank, an APC, a truck etc. or clutter.

Typically classification methodologies for autoscreening/cueing systems involve two phases. One, single computer processing which utilizes easily calculable object features such as the length, width, area, perimeter, straightness of sides etc. that provides a coarse prescreening mechanism for rejection of obvious clutter. The second phase uses more elaborate mathematical and logic operations to ascertain the object's identification. These operations may include moment calculations, Fourier or other (Walsh, Haar) transforms, logical neighborhood processing and other mathematical shape and/or intensity profile descriptors. The result is a feature vector which succinctly summarizes the object's geometric and/or radiation profile.

The feature vector is then utilized either directly against predetermined threshold boundaries or more often in a table look-up operation against a prestored set of feature vectors representing known targets. The matching process would be either a correlation or a mean square difference criterion.

MVIPS must be capable of accomplishing the above described tasks with enough expansion capability to avoid system slowdown as the number of objects under classificatin increases.

Tracking Requirements

MVIPS must be capable of implementing the traditional and new proposed approaches to the target tracking problem. It should be capable of executing centroid and correlation tracking algorithms with variable size tracking gates. In addition, noise smoothing and thresholding operations may be necessary to enhance jitter-free tracking behavior.

The last few years have seen a steady amplification of the classic tracking methodologies to encompass more general considerations that allow continuous

object tracking under more adverse operational scenario conditions. Algorithms for accommodating target obscuration and/or variable background conditions are now under development and MVIPS must be capable of incorporating them. The architectural implications of these approaches dictate that MVIPS be capable of multiple gate placement across the FOV with the algorithmic considerations possibly different for each gate. Thus, in addition to tracking the main object, MVIPS must be able to implement signature prediction for the anticipated object occupancy of areas preceding and adjacent to the tracked trajectory.

Target-Handover Requirements

MVIPS must be capable of accommodating the I/O and algorithmic requirements necessary to accomplish effective target-handoff from a primary sensor to a secondary one located on a weapon such as the Maverick missile. In effect, MVIPS must be capable of executing the target recognition function on the primary sensor data and a scene matching operation between the primary sensor scene and the weapon imagery.

The scene matching algorithms can range from a direct correlation search match to a syntactic association of segmented objects in both scenes. The fundamental considerations are no different, in principle, than those encountered in the basic target detection and classification requirements. However, the handover problem dictates multiframe storage capability and simultaneous execution of the target recognition, target tracking and scene matching algorithmic logic.

Arithmetic Computation Requirements

MVIPS must be capable of performing a variety of mathematical computations encountered in navigation and weapon delivery functions. These are basically matrix-vector operations such as state propagation, Kalman filtering and numerical integration of differential equations.

Input data can originate from a variety of sensors such as Inertial Navigation Systems, radar, air-data computers, altimeters etc. in addition to imagery data.

Figure 6 provides a brief features summary comparison of the Honeywell (PATS), Westinghouse (Auto-Q) and Ford (MVIPS/AASAD) systems.

AASAD

COMPARATIVE SYSTEMS REVIEW

FUNCTIONS SYSTEM	VIDEO PREPROCESSING	DETECTION AND CLASSIFICATION	TARGET TRACKING	COMMENTS
HONEYWELL (PATS)	• 875 OR 525 LINE VIDEO • ANALOG HARDWARE CCD DEVICES • ADAPTIVE CONTRAST ENHANCEMENT • DC RESTORATION • GLOBAL GAIN AND BIAS CONTROL	• ANALOG AND DIGITAL HARDWARE • AUTOTHRESHOLD • MOMENT FEATURES • EDGE DERIVATION • K-NEAREST NEIGHBOR RULE • INTERFRAME ANALYSIS	• PROCESSING SPEED IS ADVERSELY AFFECTED BY THE NUMBER OF TARGETS BEING TRACKED	• EFFECTIVE USE OF ANALOG AND DIGITAL HARDWARE FOR R/T PROCESSING • ALGORITHM MODIFICATIONS REQUIRE EXTENSIVE SYSTEM REDESIGN
WESTINGHOUSE (AUTO-Q)	• 875 OR 525 LINE VIDEO • DIGITAL HARDWARE • SOFTWARE CONTROL • SCENE AVERAGING • MEDIAN FILTERING • LEVEL SLICE • GRADIENT EXTRACT	• GENERAL PURPOSE COMPUTER • SCENE ANALYSIS • FEATURE MATCHING • AUTONOMOUS HANDOFF • TARGET PRIORITIZATION	• CORRELATION	• SOME PORTIONS OF IMAGE PROCESSING DONE ON GENERAL PURPOSE COMPUTER WHICH LIMITS R/T CAPABILITY
FORD AEROSPACE (MVIPS/AASAD)	• 875 OR 525 LINE VIDEO • REALTIME ANALOG ENHANCEMENT • DIGITAL VIDEO FRAME STORE • THRESHOLD/BINIARIZE • CONVOLUTION (3x3) • VIDEO EDGE MASKS • VARIABLE AVERAGING MASKS	• HIGH SPEED DIGITAL MICRO PROCESSOR • ALGORITHMS (SOFTWARE SELECTABLE) • SUPERSLICE • MOMENT INVARIANTS • STATISTICAL • SYNTACTICAL • ETC	• HIGH SPEED MICRO PROCESSOR • ALGORITHMS (SOFTWARE SELECTABLE) • CENTROID • CORRELATION • MULTIPLE TARGETS • ETC	• PRIMARILY A HIGH SPEED DIGITAL APPROACH • FLEXIBLE ALGORITHM SIMULATOR • USER ORIENTATED PASCAL BASED LANGUAGE

Figure 6. Comparative systems review

SECTION 3

SOFTWARE

ARCHITECTURE

All of the AASAD software is designed with high flexibility and ease of use with the algorithms and resource control in the Intermediate High Level Language (IHLL). The software is easily expanded by generating additional Macros and Micros for inclusion in the libraries. Maintenance of the system is reduced by having all of the software in small modular routines. The system utilizes a Pascal based Intermediate High Level Language that is resident in the System Controller. The IHLL allows the user to program the AASAD system in a high level language which is much easier than programming at the macro and micro level. Through the use of IHLL, a user is able to control the serial and parallel functions of AASAD.

AASAD is user oriented and allows the user to control particular algorithmetic functions within the individual processors or to operate on complex algorithms. The algorithms are generated in the IHLL and loaded into the System Controller.

The System Controller generates control block lists which are transferred to the Control Processor. The Control Processor directs the configuration controller, Graphics Controller and High Speed Digital Microprocessor as shown in Figure 7. It can thus be seen that AASAD is a list driven distributive processing system. The relationship between the software and hardware is also shown in Figure 7.

Based upon the function to be performed, various microcomputers and processor of various word sizes are utilized as shown in Table 1.

SOFTWARE DEVELOPMENT

At FACC, all AASAD software is developed on the VAX 11/780 or the PDP 11/23. Both computer systems are part of the Digital Systems Department Image Processing Laboratory. The PDP 11/23 is essentially the same computer as the LSI 11/23 (the System Controller). The PDP 11/23 has additional peripherals (disk, Decwriter, etc) that allow FACC to use it as a software and hardware development computer. The VAX 11/780 is a large minicomputer with 3 megabytes of memory, modems, display terminals, line printer and 400 megabytes of disk storage. The computer system is used as a multiple user work station allowing the users to develop IHLL, Macros and Microcode simultaneously.

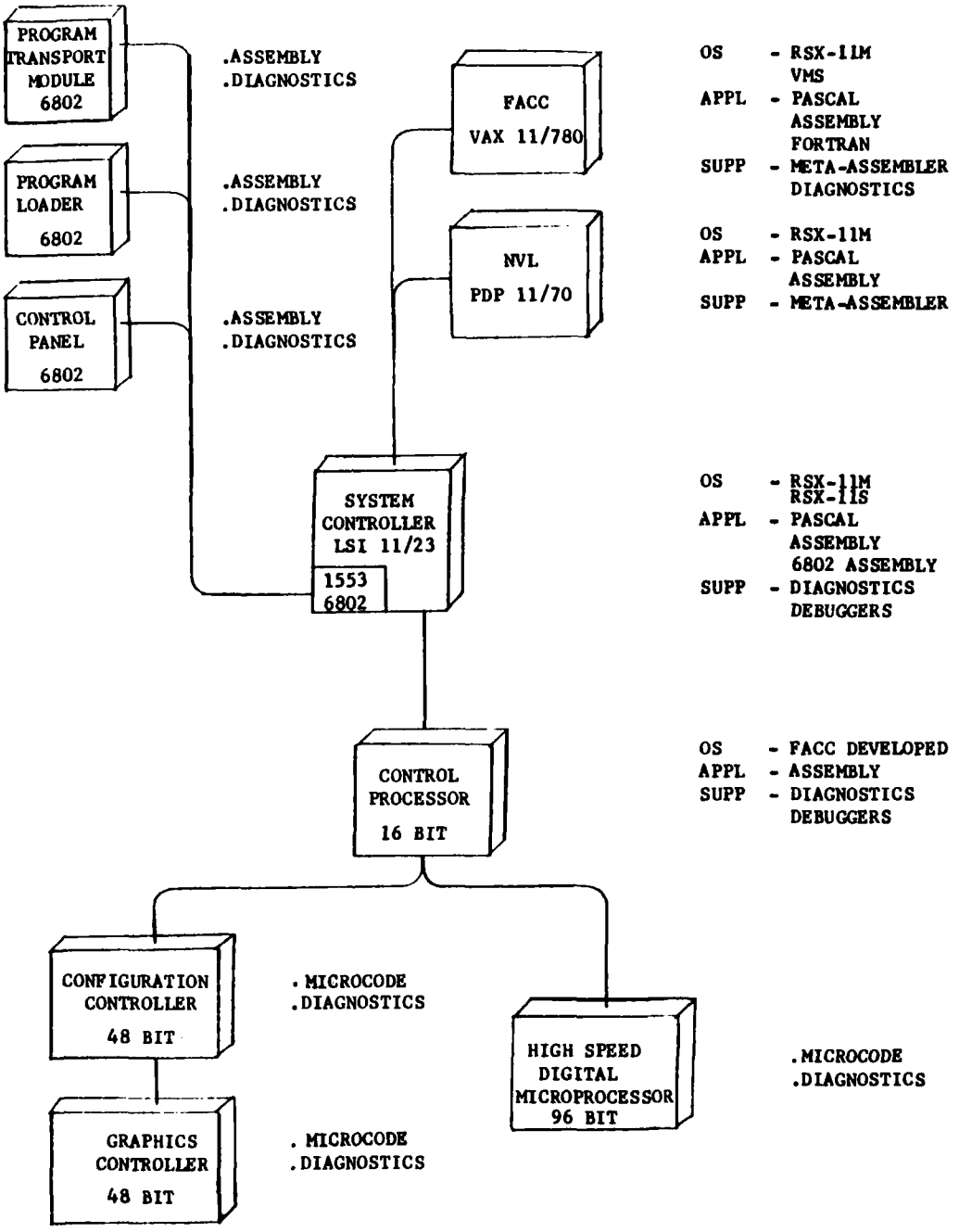


Figure 7. AASAD software

TABLE 1. PROCESSORS

	Word Size
• Motorola MC6802	8
- Microprocessor	
- Program Transport Module	
- Program Loader	
- Control Panel	
- 1533 Interface to LSI 11/23	
• VAX 11/780 (Compatibility Mode)	32 (16)
PDP 11/70	16
• Control Processor	16
• Configuration Controller	48
• Graphics Controller	48
• High Speed Digital Microprocessor	96

Both of the computers utilize the DEC RSX-11M operating system. In addition to the usual support utilities supplied by DEC (compilers, assemblers, editors etc), the Oregon Software Pascal and the Step Meta-Assembler are installed on both of the computers. The Oregon Software PASCAL is used to develop the IHLL and PASCAL Macros. The Meta-Assembler is used to generate the micro-code for the Configuration Controller, Graphics Controller and High Speed Digital Microprocessor.

All support software except for licensed software and the application software (micro code) for AASAD is directly compatible with the NVL PDP 11/70 computer system without any modification.

SOFTWARE SUPPORT

FACC recognizes the need for good support tools and diagnostics and the cost of developing these tools. Whenever possible, commercially available software is utilized. For the VAX 11/780, FACC utilizes the full capability of DEC supplied software tools and diagnostics. Additional purchased software packages are a Fortran assembler for the Motorola MC6802, Step Engineering Meta-Assembler and the Oregon Software Pascal Compiler. On the System Controller (LSI 11/23), FACC is utilizing the DEC supplied RSX-11S operating system, diagnostics and utilities.

FACC is developing unique software tools only when required. Some of the tools and utilities include a software debug Control Panel program for the

Configuration Controller, Graphics Controller and HSDM. Through the System Controller, the user is able to single step a program, change addresses and set breakpoints. Also diagnostics are being developed for all of the processors, memories and interfaces on AASAD. The diagnostic software is modular in design, and written in PASCAL whenever possible. In addition to the individual processor diagnostics, FACC is developing a system diagnostic that allows a user to test and exercise the total system.

The hierarchy of the AASAD software is as follows:

- Intermediate High Level Language (IHLL)
- Pascal Macros
- Microcode

INTERMEDIATE HIGH LEVEL LANGUAGE (IHLL)

To ensure continued growth and expansion of the AASAD software, FACC has simplified the implementation of algorithms and the software development process by utilizing a Pascal based Intermediate High Level Language. Figure 8 shows the typical software development process for the IHLL. The advantages of the IHLL are as follows:

- User Oriented
- Intermixes with Oregon Software Pascal
- Allows transfer of data/results to and from the AASAD system
- User interface to the AASAD system

The IHLL is a user oriented version of the Oregon Software Pascal base. The Pascal compiler is user oriented through the utilization of a selected list of Pascal procedures. The IHLL is developed on the NVL PDP 11/70 or FACC VAX 11/780. The IHLL will be executed in the System Controller of AASAD. The development process will utilize Pascal generated macros that will utilize selected micros to perform the algorithm defined by the macro. The macro development process is depicted in Figure 9.

PASCAL MACROS

The macros are generated in relocatable Pascal and are executed as part of the IHLL in the System Controller. The macros are the executable part of the IHLL and the macros generate the control block lists that are transferred to the Control Processor. As shown in Figure 9, all of Pascal routines are combined together to form the Macro Library which is utilized by the IHLL. The Pascal macros are developed on the PDP 11/70 or VAX 11/780.

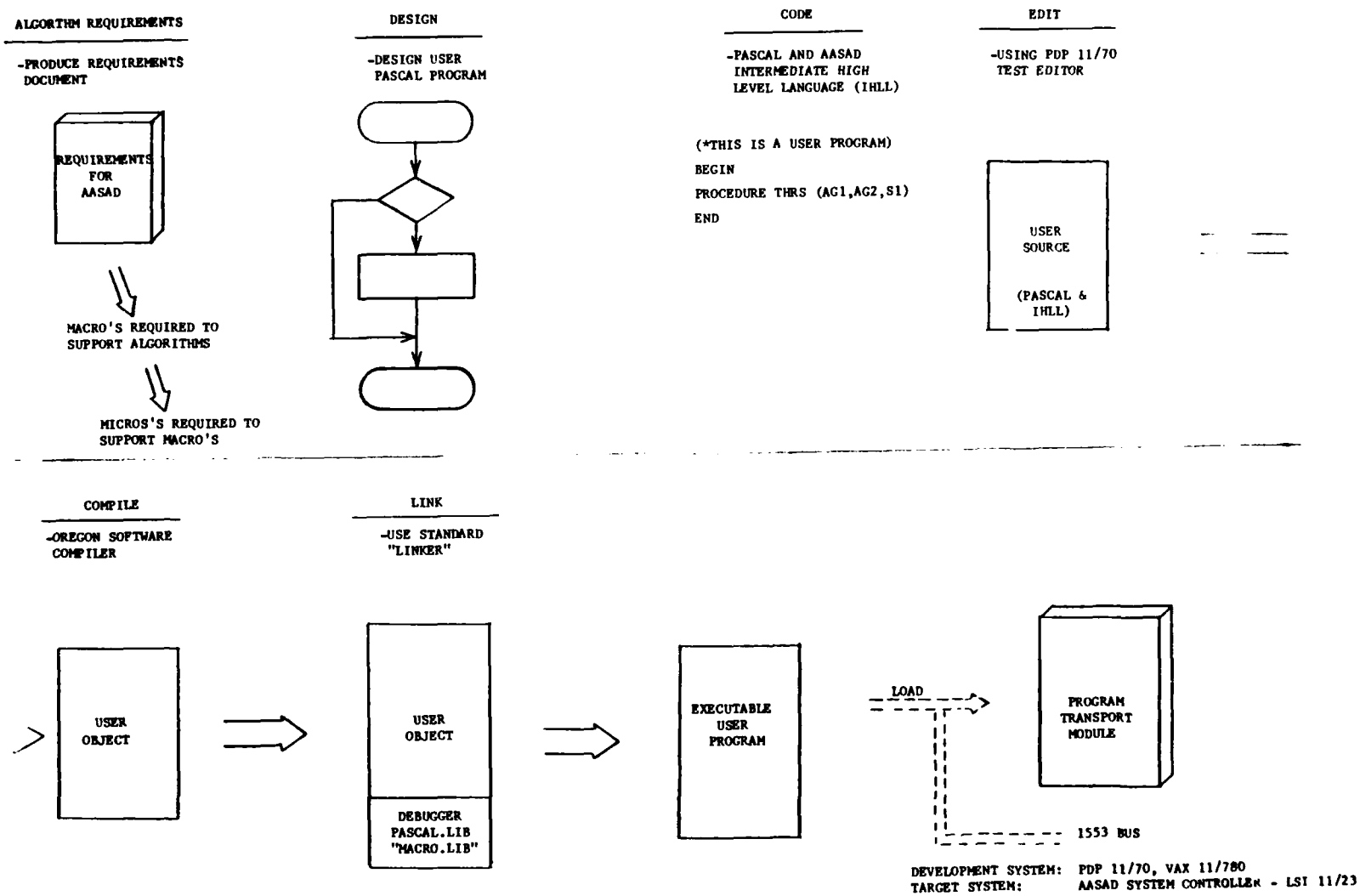


Figure 8. Intermediate high level language (IHLL)

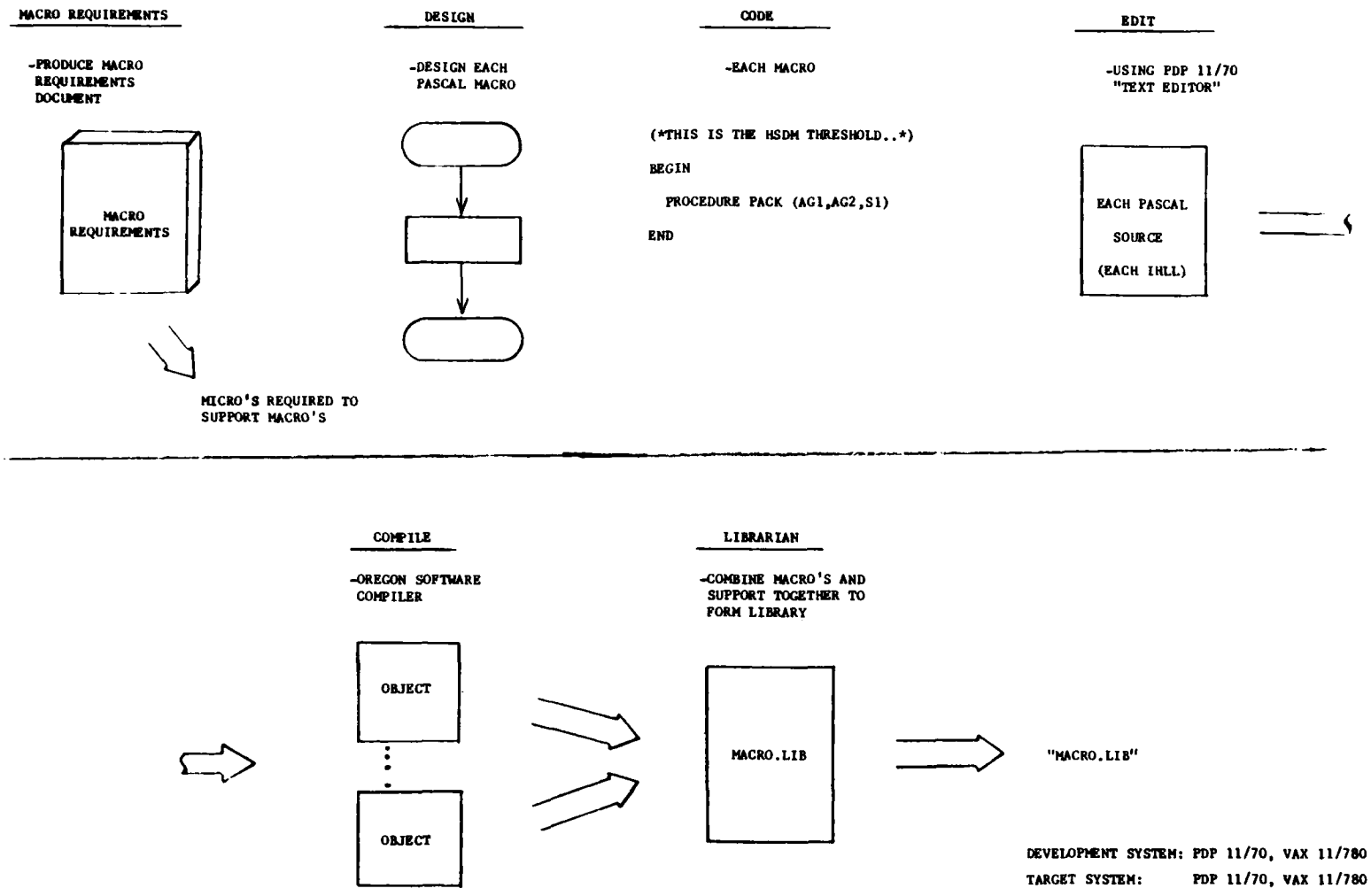


Figure 9. Macro development

MICROS

The algorithm microcode, generated in assembly language, utilizes absolute addressing. Development of the microcode is also on the PDP 11/70 or VAX 11/780 and the execution of the microcode is in the following microcomputer:

- Configuration Controller (48 bits)
- Graphics Controller (48 bits)
- High Speed Digital Microprocessor (96 bits)

The microcomputers process the image data and execute the algorithms as commanded by the Control Processor. It is at the microcode level that the AASAD system attains its near real time capability. The microcode development process is shown in Figure 10. An example of a typical microcode routine is shown in Figure 3-39 of the Appendix.

OPERATION OF THE SYSTEM

The AASAD software is developed on the PDP 11/70 or VAX 11/780 and loaded into the AASAD system through the MIL-STD-1553A bus or the program transport module as shown in Figures 8, 9, and 10. The executable software is then loaded into the individual microcomputers memories as shown in Figure 11.

The operation of the AASAD system is shown in Figure 12. The system is loaded and initialized by using the Program Transport Module or 1553A bus. The System Controller executes the IHLL intermixed with standard Pascal. When an IHLL instruction is executed, the System Controller executes the appropriate Pascal macro which sets up the proper control block list or lists. The lists are then transferred to the Control Processor. The Control Processor has a two pass scheduler where the first pass fills in any parametric data need by the control block lists. On the second pass, the scheduler will either execute the list immediately by sending the list to the proper processor or queue the list for execution when the processor is available.

The AASAD Executive, operating under the RSX-11S Operating System within the System Controller, consists of a series of PASCAL procedures, bootstrap routines, the AASAD utilities and AASAD application software. The utilities are summarized as follows:

(1) Buffer Configuration: The buffers are assigned, organized, and dedicated to specific locations within the Control Processor (CP) and other processors.

(2) Debug Routines: Routines that will assist in the debugging of software within the System Controller (SC) which includes status and parameter dumps from the Control Processor.

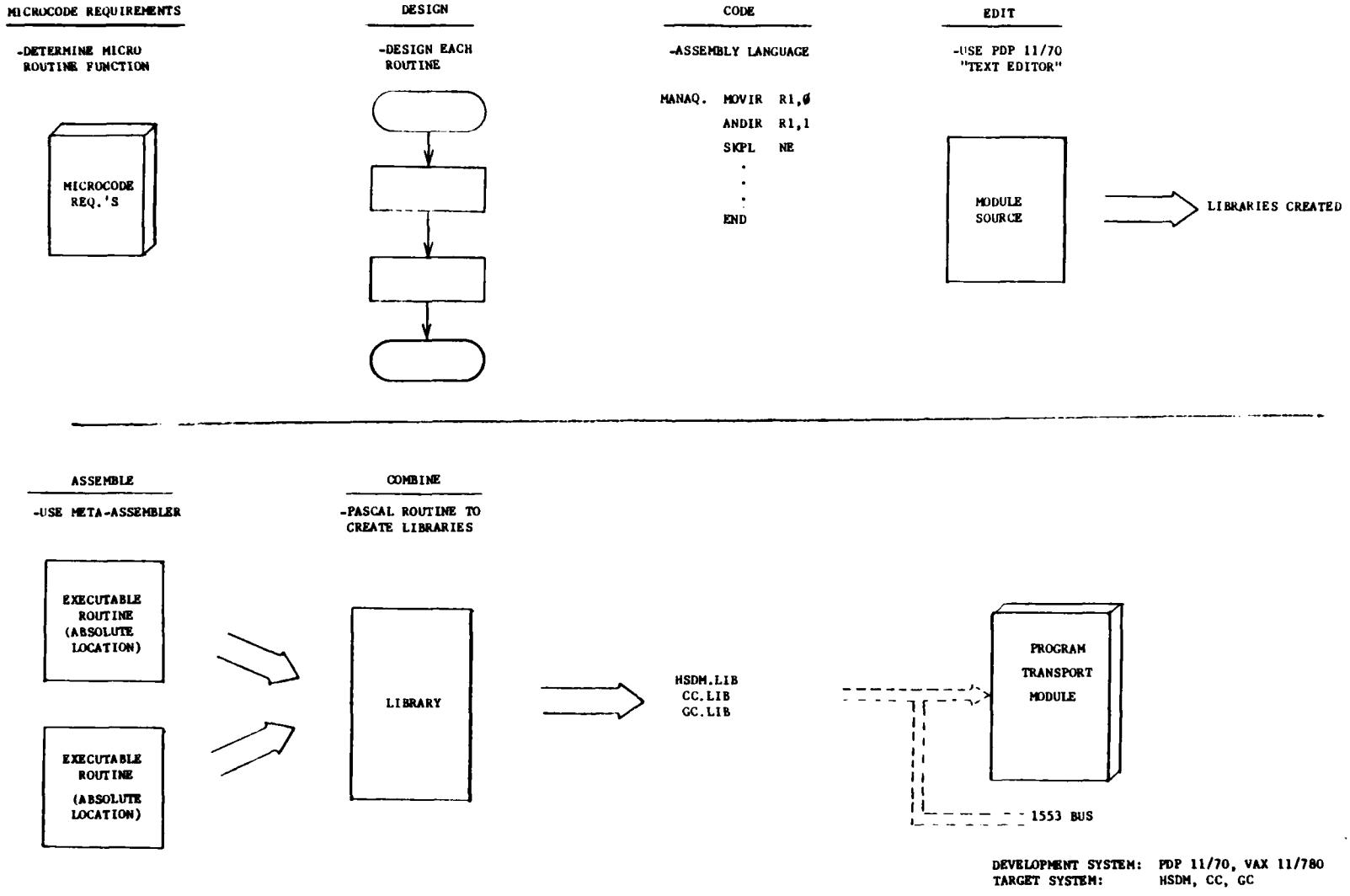


Figure 10. Microcode development

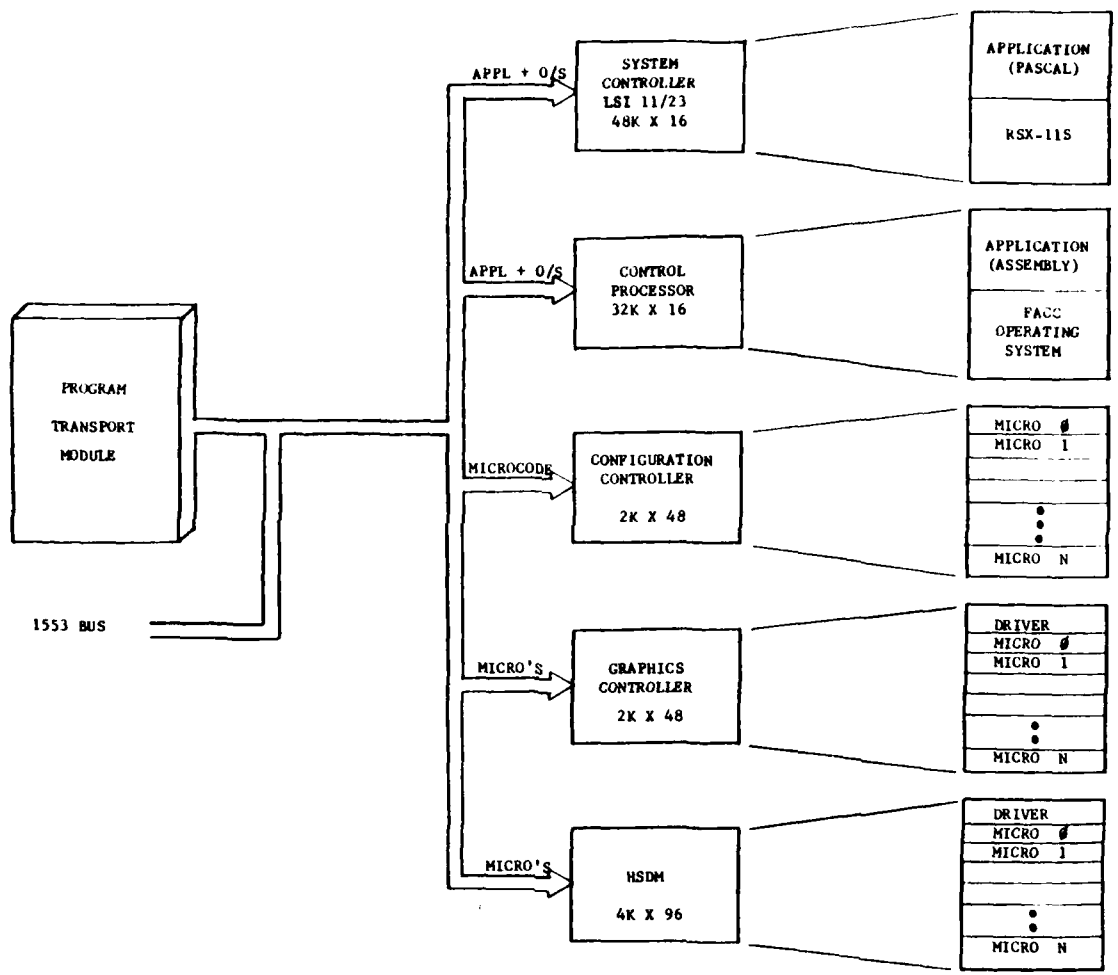


Figure 11. Initialize AASAD

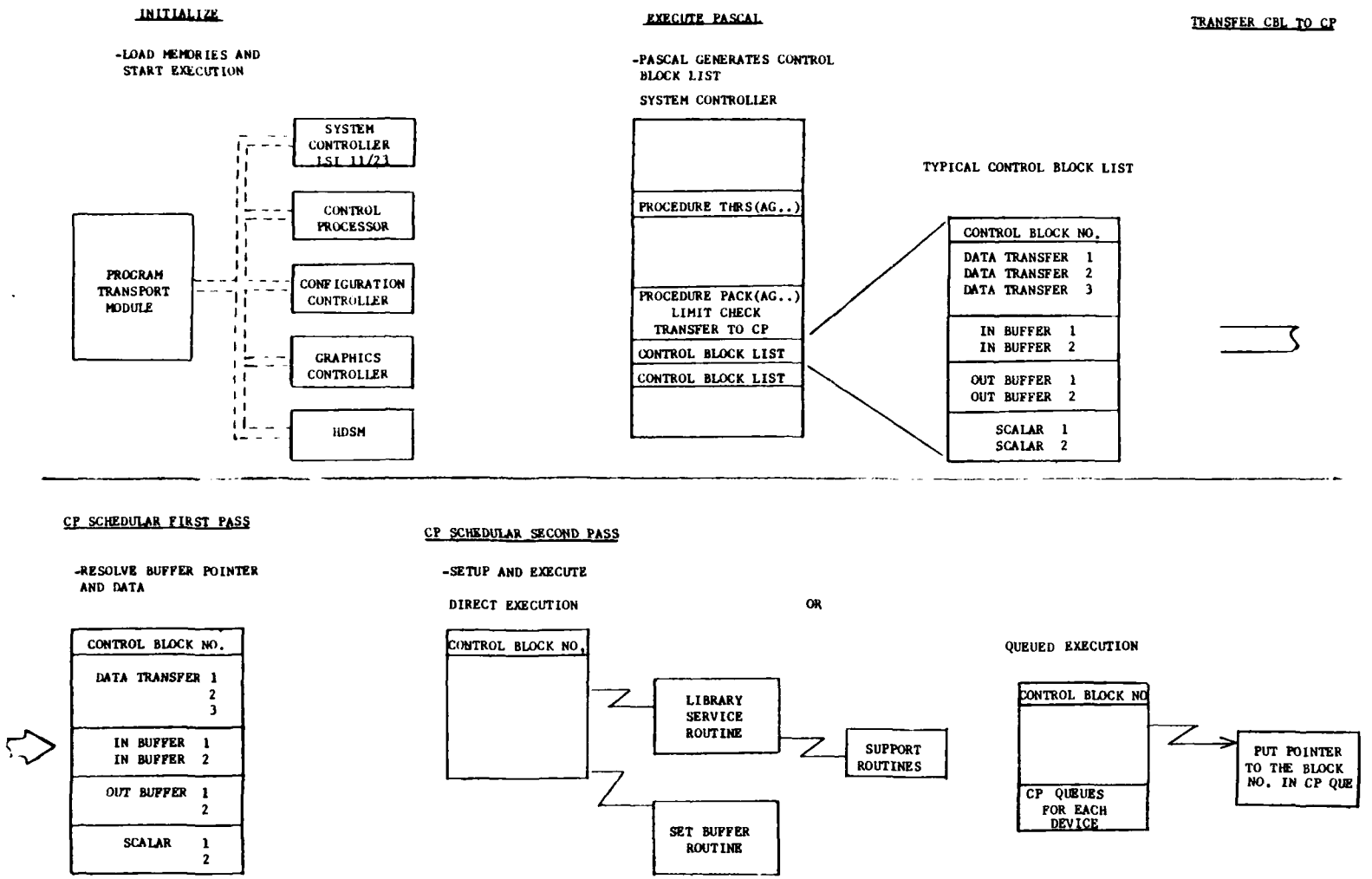


Figure 12. Execution of IHLL in AASAD

(3) Diagnostics: Diagnostics are loaded into the Control Processor from the System Controller.

(4) Data Transfer Routines: These are parameters and I/O routines.

(5) Math Library: Specifies mathematical routines that are necessary in the System Controller to support the Control Processor.

(6) List Routines: Routines that are necessary to generate and execute Control Processor lists.

The AASAD or Control Processor Operating System (OS) will be a realtime, multitasking, multiprocessing, list driven operating system. Its primary function will be the resource allocation and control of the video image processing as commanded by the System Controller.

The CP OS commands can be classified into four principal categories as follows:

(1) Command Sequencing. This class of commands allows the user to define the sequence of tasks performed in the image processing. For example, continue acquisition processing until a target detection; after which a classification is performed.

(2) Task Execution. These commands identify which application software is to be executed and the conduct of execution, e.g., binarizing an image within a predetermined gate.

(3) Resource Allocation. The resource allocation commands provide the user with control over data flow and a limited AASAD processor interface definition. For example, video image data flow from the Video Frame Memory to the Pipeline Arithmetic Processors or for video line by line processing control over the variable matrix preprocessor.

(4) Macro Instruction Execution. The macro instructions are contained within a Macro Instruction Library and represent a definite sequence of commands relating to each macro. The macro instructions are a predefined sequence of the basic utility commands identified by the other three categories. An example of a macro command is outlined as follows:

- Geometric Centroid Macro
  - Move data to Pipeline Arithmetic Processor (Resource Allocation)

- Move data from PAP cache memory to scratch pad for processing (Task Execution)
- Limit Test data (Task Execution)
- Binarize Limit Test output (Task Execution)
- Compute Matrix Sums (Task Execution)
- Move Sum output to Control Processor (Resource Allocation)
- Compute Centroid from matrix sums (Task Execution)

SECTION 4

HARDWARE

DIGITAL VIDEO PREPROCESSORS (DVPP)

The AASAD systems initial complement of video preprocessors consist of four functions on three cards:

(1) Sum/Difference and Threshold/Binarization

(2) Variable Averaging Mask

(3) Video Edge Detection Mask

A fourth card is required to interface the preprocessors to the AASAD internal BUS system. The Preprocessor Interface Module (PPIM) also performs the translation between the high-speed Emitter Coupled Logic (ECL) of the DVPPs and the Transistor/Transistor Logic (TTL) circuits of the other modules on the BUS systems. Figure 13 is a simplified block diagram showing how the PPIM interfaces with the DVPPs and the AASAD BUS system.

The PPIM interface and the Configuration Controller is depicted in Figure 14 and consists of the following signals:

CNFD	- 16 Lines	-	High True	Bidirectional
CNFA	- 5 Lines	-	High True	Unidirectional
R/W	- 1 Line	-		Unidirectional
CREN	- 8 Lines	-	Low True	Unidirectional
DISC	- 1 Line	-	High True	Unidirectional
CNFSTB	- 1 Line	-	Low True	Unidirectional

These commands are used to configure the PPIM. Configuration of the PPIM requires ID and IE Bus data source selection, ID and IE Bus blanking source selection, and configuring the Area of Interest (AOI). See Figure 15.

Configuration of the PPIM is performed during vertical blanking by a burst transfer from the Configuration Controller.

The FIFO is reset, then loaded with 64 words during each vertical blanking. The reset is in hardware - not software control. The first word is the line count defining the first line of an AOI. The line counter is incremented at the start of horizontal blanking. When this count equals the line count from the FIFO, the next words from the FIFO, (which may contain up to 16 start and stop pixel addresses), are written into the 16 x 12 RAM. This transfer to RAM stops when the next line address is detected from the FIFO. This operation is asynchronous and takes less than 2 microseconds.

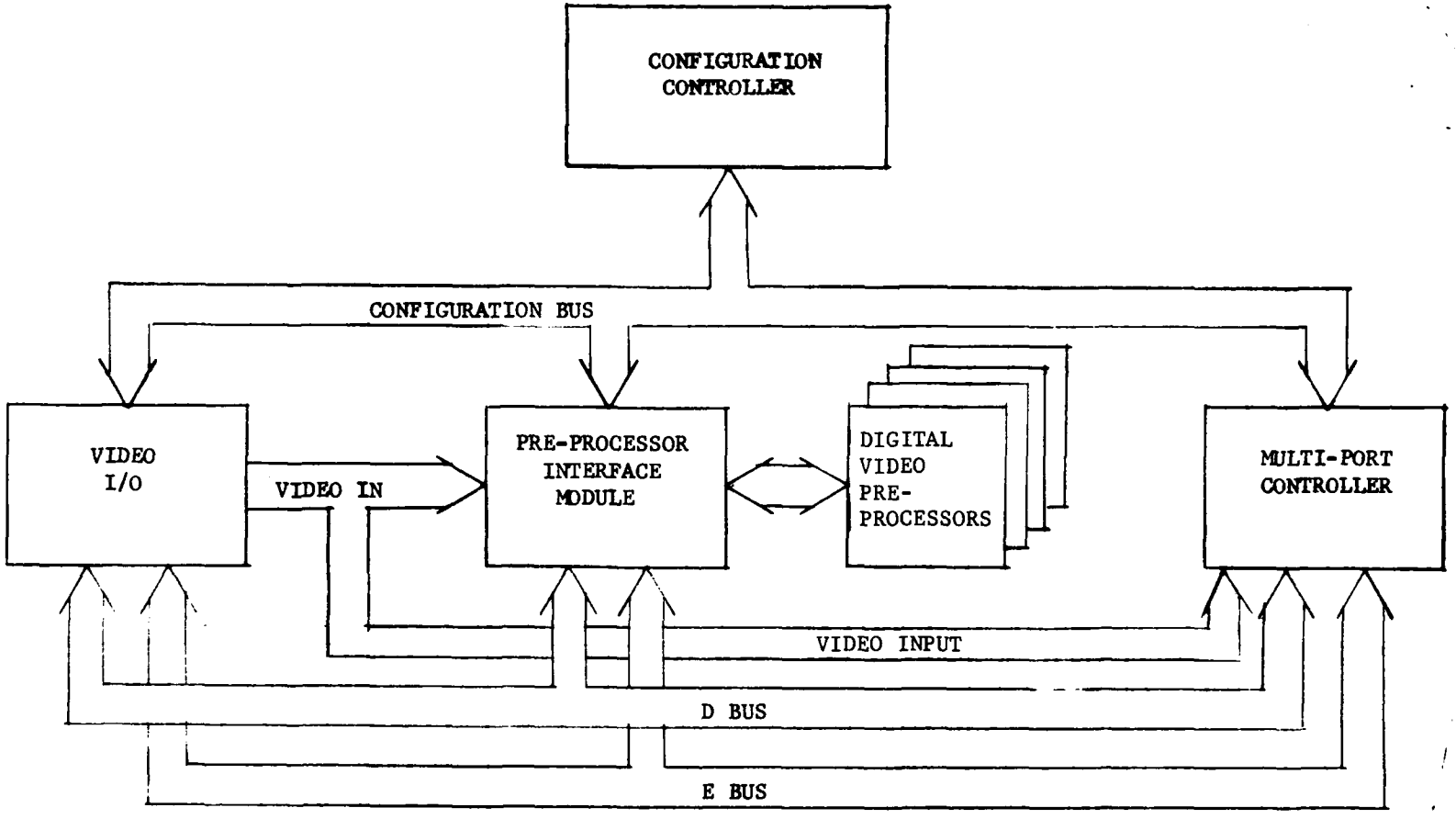


Figure 13. Digital video preprocessor simplified interface block diagram

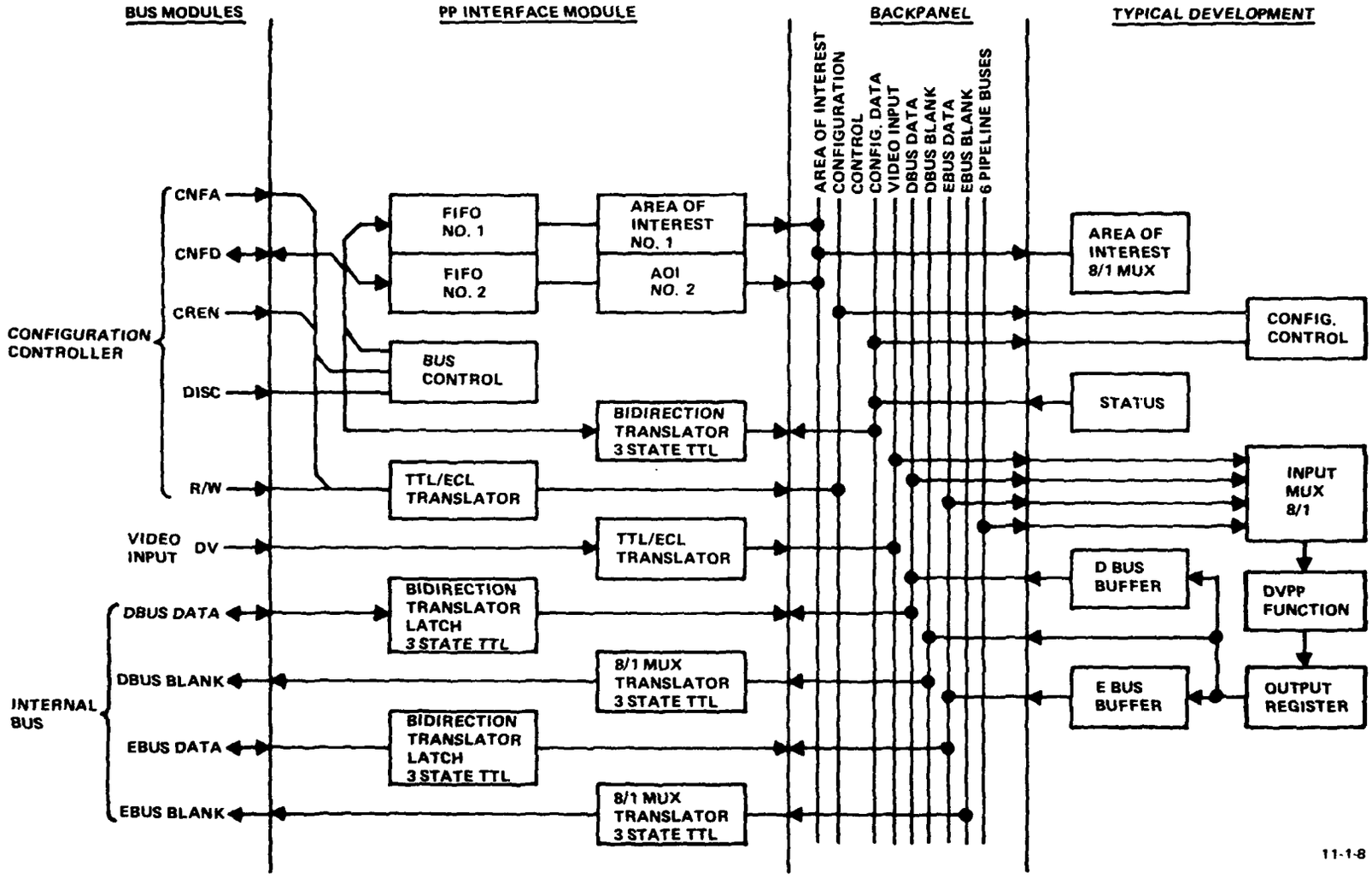


Figure 14. PPIM interfaces

DISC - Hi True CNFA - Hi True
CREN - Lo True CNFD - Hi True
CNFSTB - Lo True

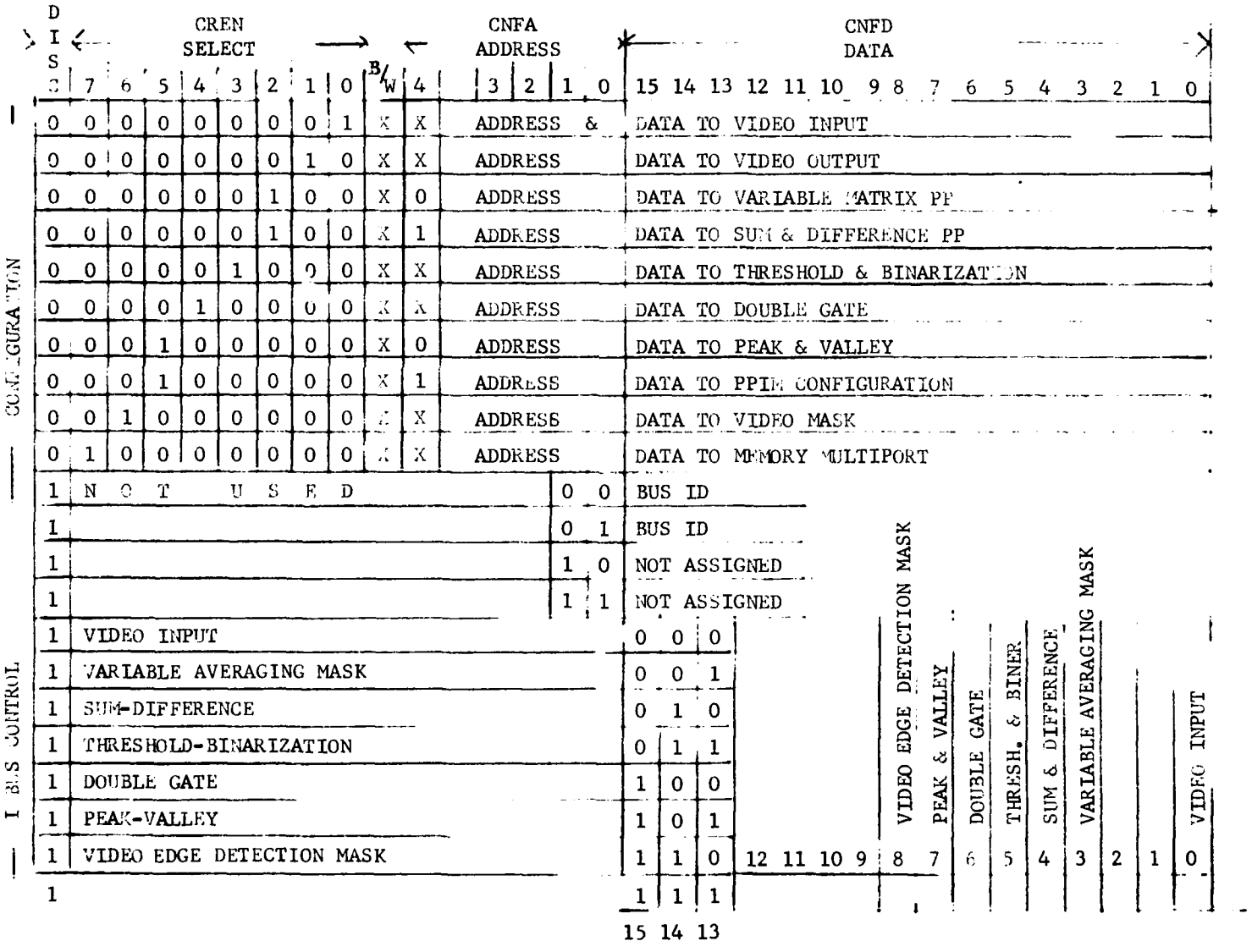


Figure 15. Address and control matrix

During line time the pixel counter is compared with the first word in memory which is the first start pixel address. A comparison sets the Run condition and increments the 4 bit counter to the second RAM address which contains the stop pixel count. A comparison stops the AOI and increments the 4 bit counter to the third RAM address which contains the next start pixel address, etc. Figure 16 is a functional diagram of the AOI logic.

The 4 bit counter and the pixel counter are reset at horizontal blanking time. The operations described above will be repeated for each line until the next line count comparison is made and the RAM is reloaded with another set of start/stop pixel addresses.

Configuration Data Definitions:

Configuration Data (CNFD) bits 0 → 9 are the pixel/line addresses.

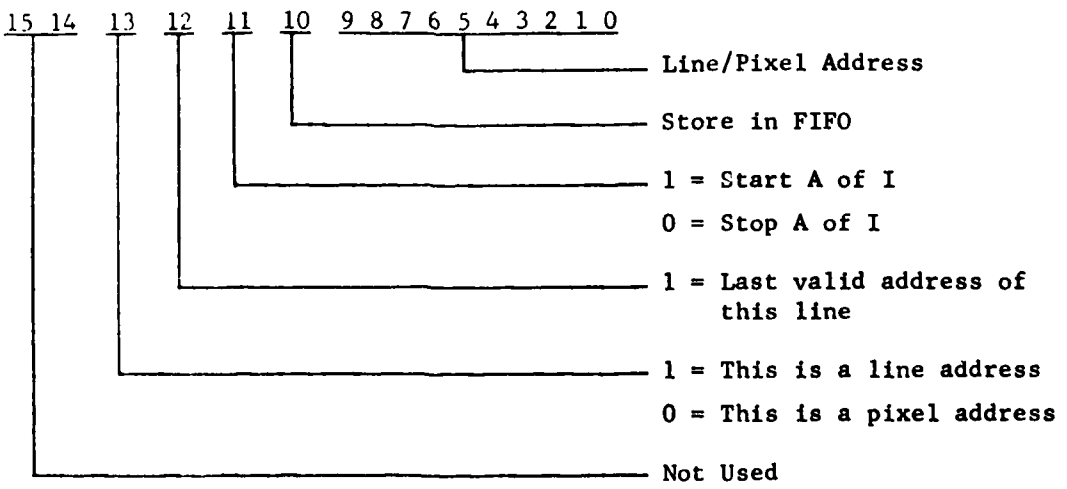
CNFD 10 directs this word to the FIFO. This bit is not loaded into the FIFO.

CNFD 11 when set defines the address (CNFD 0 → 9) as the "Start AOI" address. When clear it is the "Stop AOI" for address.

CNFD 12 when set defines this address as the last instruction of the line. It inhibits the 4 bit counter (Figure 16) and so prevents old data from the RAM.

CNFD 13 when set defines CNFD 0-9 as the line address. When clear, CNFD 0-9 is the pixel address.

CNFD 14, 15 are not used.



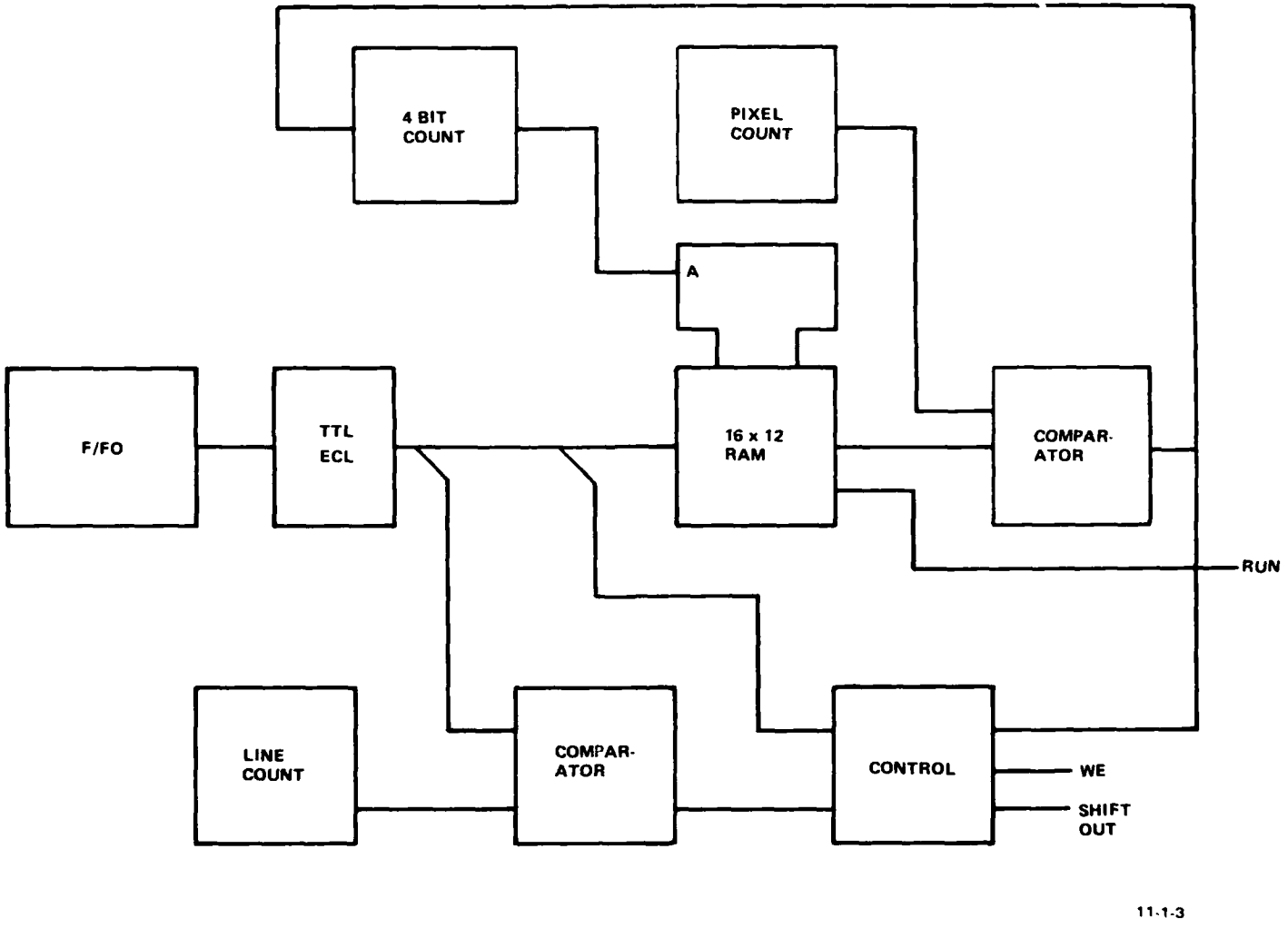


Figure 16. AOI functional logic

ID and IE buses each contain 16 bidirectional data lines, two unidirectional blanking lines, and two unidirectional Clock Enable lines. The least significant 8 data lines are defined as byte 0, and the most significant 8 data lines are byte 1. Pixel values are transferred on alternate bytes always starting a line with byte 0. This alternating transfer is to allow more time for each value to be transferred.

Data transferred from memory contains configuration data with each value on its lines for 2 clock periods. The PPIM latches byte 0 on the first rising edge of the clock after blanking goes false. The next value is latched is byte 1 on the next rising edge of the clock. This alternating continues until the end of the line and blanking goes true again. Clock Enable lines are not used when taking data from memory.

Data to memory is also transferred as alternating bytes. However, they are not necessarily contiguous. To accommodate this, two Clock Enable lines (Bytes 0 and 1) are provided. These lines go true when their associated data bytes have been on the lines for one clock period and stradle the next rising edge of the clock at the Multiport Controller (MPC). This allows about 1-1/2 clock cycles for propagation delay and setup time and the difference in the clock phase between the PPIM and the MPC.

Sum/Difference and Threshold/Binarization DVPP

The Sum and Difference (S&D) determines either the average of two pixel values or the absolute difference between two values as specified by the programmed configuration. When configured to sum, the two values are added together and the sum is then scaled right one place (divide by 2) so the output is actually the average of the two input values. When configured to determine the difference, the two input values are compared for magnitude, the lesser value is converted to its 2s complement, the negative value and is added to the greater input value.

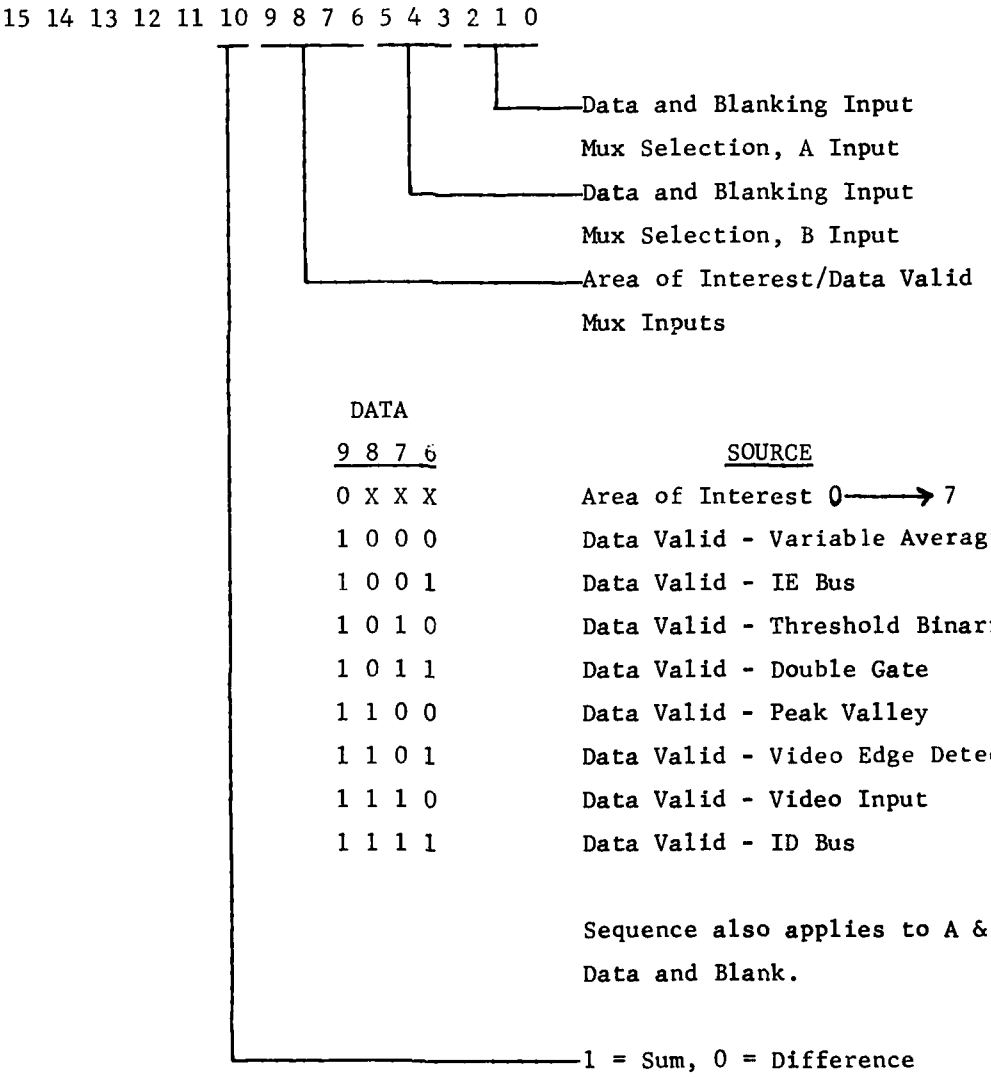
Input data to the S&D logic comes from two sources as defined by configuration. When this filter is comparing the current video in field, with a previously stored field from memory any output will identify movement within the field. This assumes that the platform for the camera/FLIR is stationary or that there is compensation for platform motion between the fields. When the S&D is configured to sum (average), it becomes a smoothing filter.

Threshold and Binarization (T&B) (see Figure 17). The purpose of the T&B logic is to determine if the value of each pixel is within established limits. The value of the incoming pixel is compared with the values set in the upper

S & D CONFIGURATION

CNFA ADDRESS 0000

CNFD DATA BITS



DATA 9 8 7 6	SOURCE
0 X X X	Area of Interest 0 ⟶ 7
1 0 0 0	Data Valid - Variable Averaging Mask
1 0 0 1	Data Valid - IE Bus
1 0 1 0	Data Valid - Threshold Binarization
1 0 1 1	Data Valid - Double Gate
1 1 0 0	Data Valid - Peak Valley
1 1 0 1	Data Valid - Video Edge Detection Mask
1 1 1 0	Data Valid - Video Input
1 1 1 1	Data Valid - ID Bus

Sequence also applies to A & D Input Data and Blank.

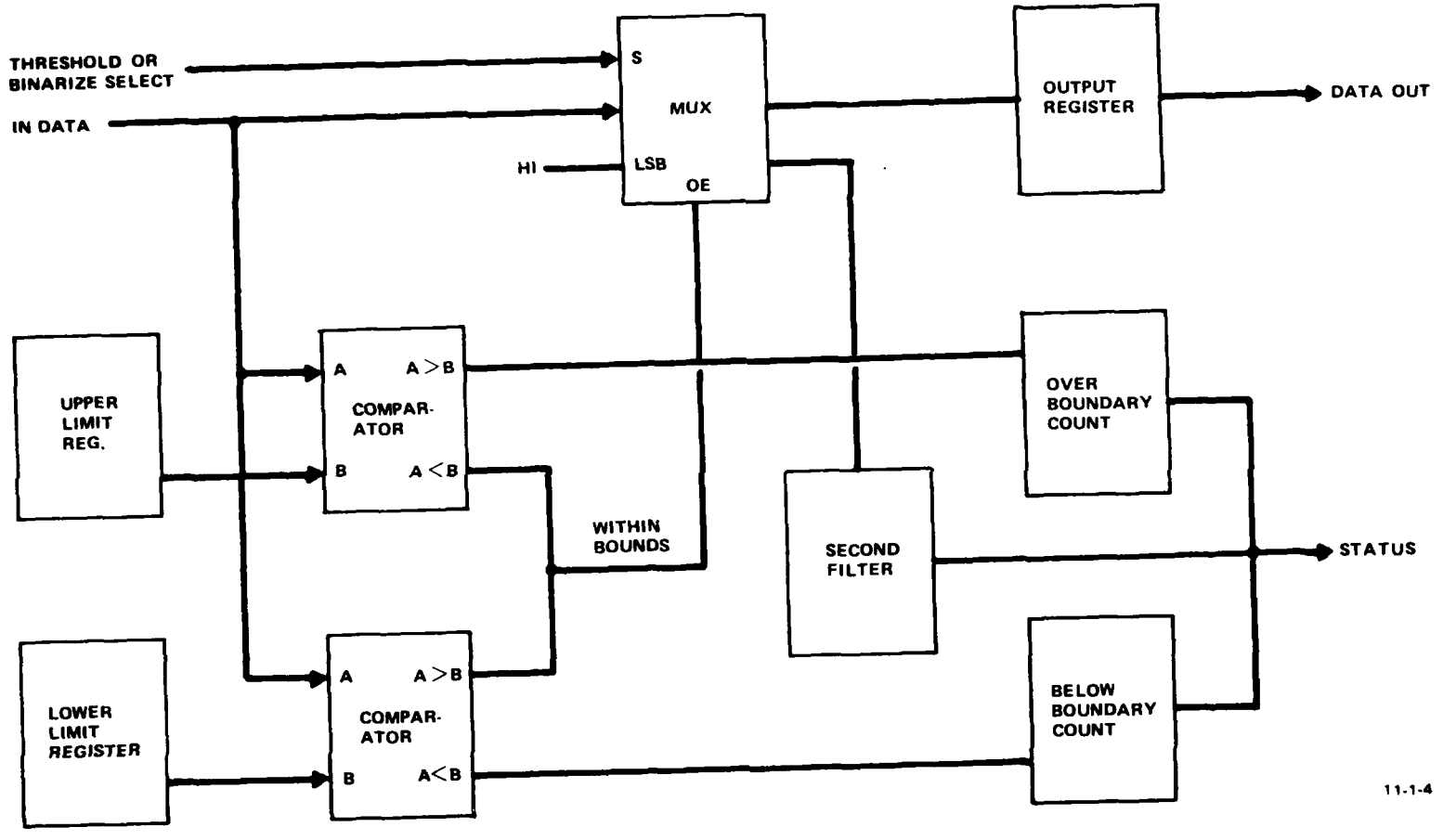


Figure 17. Threshold and binarization

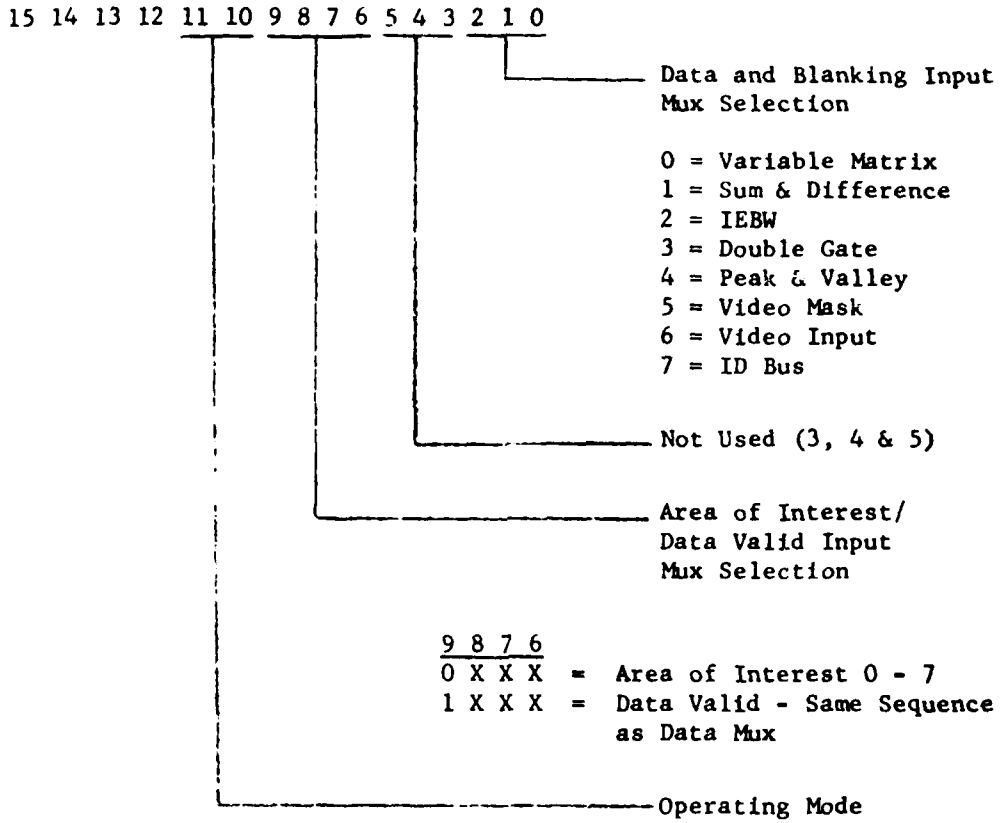
threshold and the lower threshold registers. If the pixel value is between these threshold values, either the pixel value or a "1" in the LSB is output. If the pixel value is above the Upper Threshold or below the Lower Threshold, a "0" is output and either the Above Threshold counter or Below Threshold counter is incremented. The Configuration Controller may interrogate the counters as status words.

This DVPP contains a second filter that looks at the LSB of the T&B PP output. When the T&B PP is configured to binarize the output, a "1" LSB states that the pixel value is in bounds. The LSB is run through a six-bit shift register at pixel clock rates so the six outputs contain the binarized data on the current and the five previous pixels. These six bits become six of eight addresses to inputs of a 256 x 4 PROM. The two other address bits come from the configuration register. Two outputs from the PROM are used to set or clear a flip flop. A change in state of the flip flop enters the current pixel address into a FIFO which can be interrogated by the configuration controller.

The purpose of this second filter is to alert the Control Processor (via the Configuration Controller) to possible targets. This filter will detect periods when the pixel values are within or outside the threshold limits.

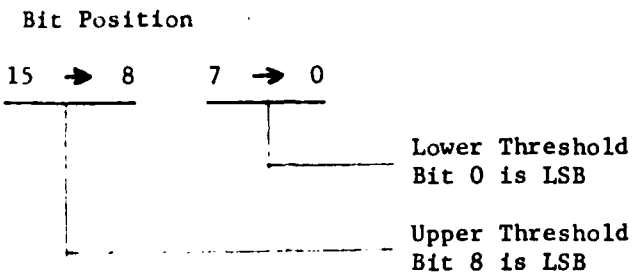
Data Word at Address 0000

Bit Position



Data Bits		Output Data When Input Is:					
		Below		Within		Above	
11	10	Threshold		Threshold		Threshold	
0	0	0	0	Data		0	0
0	1	0	0	0	1	0	0
1	0	0	0	Data		F	F

Data Word of Address 0001



Status Words

R/W line set to Read

Address 0 0 1 0 Data 9 → 0 = Over Threshold Count
Address 0 0 1 1 Data 9 → 0 = Below Threshold Count
Address 0 1 0 0 Data 11 → 0 = Second Filter FIFO Out

Variable Averaging Mask

This DVPP determines the average value of the pixels within each mask specified by the Configuration Controller (CC).

Individual masks are configured during field/frame time rather than during vertical blanking time. Each change in the mask must be configured prior to the end of horizontal blanking preceding the first line of the mask. Configuration consists of defining the number of pixels and the number of lines in the mask set. The number of pixels may be 2, 4, 8, 16, 32 or 64. The number of lines may be 1, 2, 4, 8, 16, 32 or 64. Once configured, the DVPP begins operation when enabled by the Area of Interest (AOI).

Figure 18 is an example of a field where the AOI is Pixels 22 → 37 and lines 9 → 19. Prior to the end of line 8, the DVPP is configured for the first mask of 2 pixels by one line. During line 9 the average values of pixels 22 and 23, 24 and 25, 26 and 27, etc., will be output. Before the end of line 9 the DVPP is reconfigured for 2 pixel by 2 line masks. During line 11 the average values for pixels 22 and 23, lines 10 and 11; Pixels 24 and 25 - same lines, etc., will be output. Prior to the end of line 11 the mask is reconfigured for 4x4 pixels, etc.

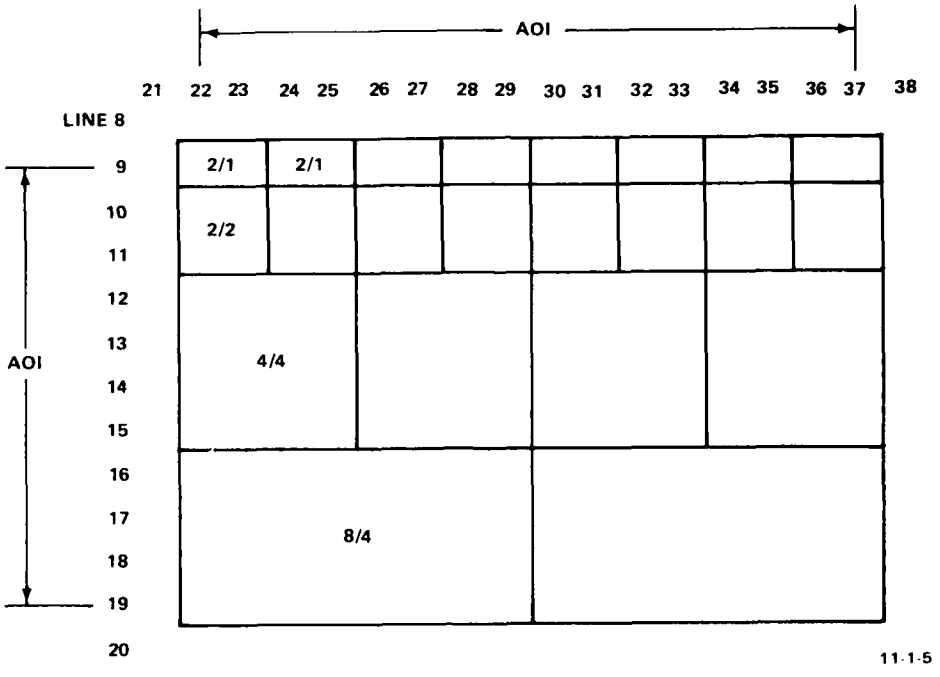


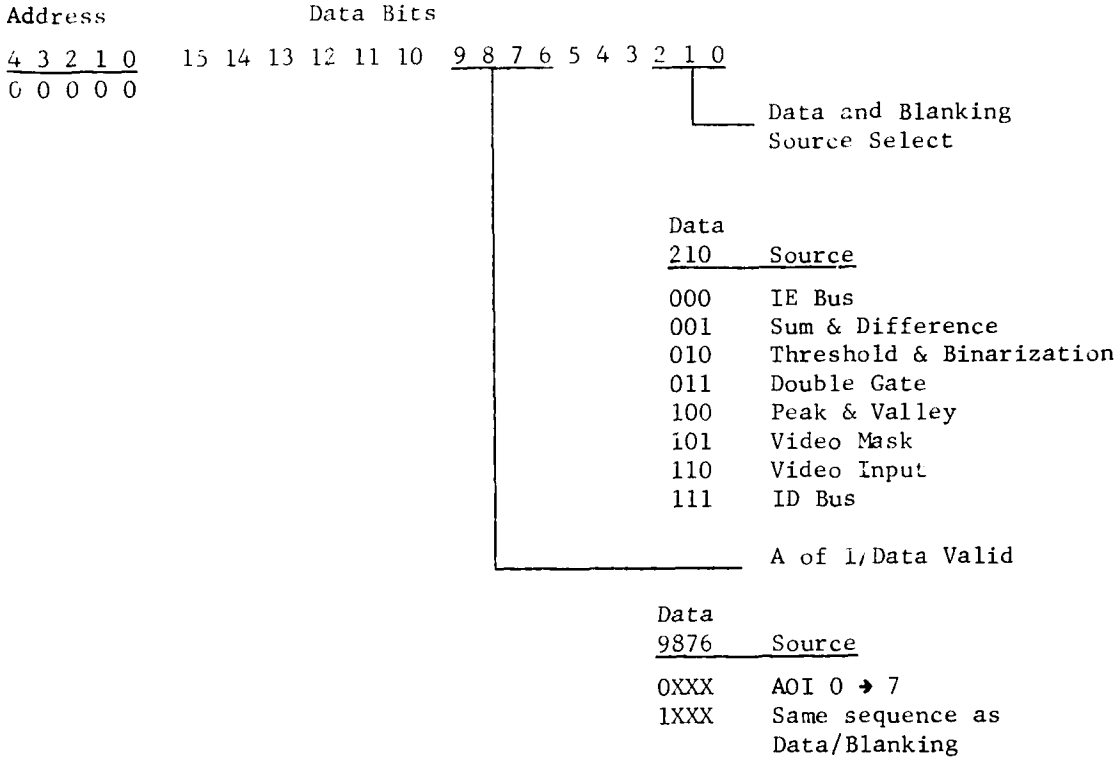
Figure 18. Variable averaging mask

It is not necessary to reconfigure if a mask set is to be repeated. If the 2x2 mask is to be repeated for lines 12-13 and 14-15, no action by the CC is required during lines 11 or 13.

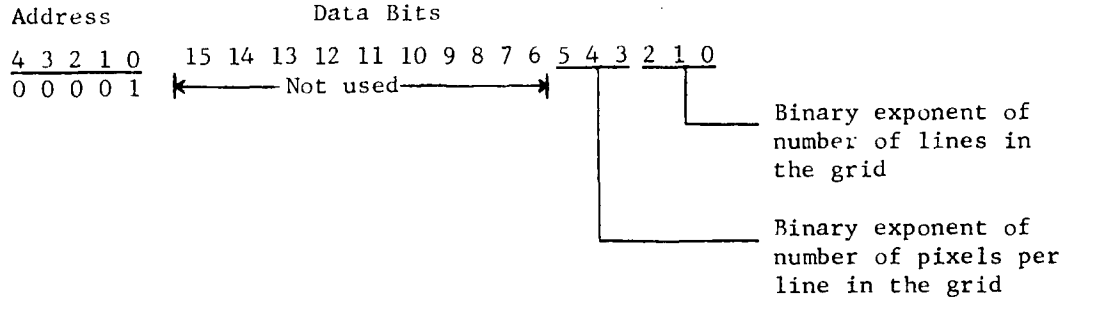
The average pixel value within each mask is determined by accumulating the sums of pixel values in each mask and when the last pixel value of the last line of each mask is detected and its value placed into the summing register, the contents of the summing register is scaled by the required factor.

Figure 18, pixel 23, line 9 is the last pixel of the last line of the first mask in that set. Since the sum of 2 pixel values is in the Summing Register, its output will be scaled right 1 place to divide the sum by 2. At pixel 25, line 15, the sum of 16 pixels will be scaled right 4 places to divide by 16. Determination of the scaling requirements and its execution is in hardware and not under configuration control.

Variable Matrix Source Configuration



Grid Dimension Configuration



This Grid Dimension must be entered before the end of Horizontal Blanking preceeding the first line of the grid. The dimensions of the grids in the Variable Matrix are configured by setting the binary exponent of each dimension in the configuration register. The limits for pixels are 2 and 64 ($2^1 \rightarrow 2^6$) and the limits for lines are 1 and 64 ($2^0 \rightarrow 2^6$).

Video Edge Detection Mask

The Video Edge Detection Mask (EDM) Preprocessor performs arithmetic operations on the nine pixel values of a 3 x 3 matrix. Consider this mask as a window on a TV raster, 3 lines high and 3 pixels wide, and moving so the current pixel is always in the lower right-hand corner. Thus, each pixel time (after the third pixel of the third row) three new pixels enter from the right and three drop off on the left.

The purpose of this preprocessor is to function, under program control, as a variety of video masks. Examples of some video masks are shown in Figure 19. To accomplish this, the nine pixel values of the 3 x 3 mask pass-through scalers and into an arithmetic pyramid. Here, under program control, the necessary arithmetic operations are performed (see Figure 20).

The EDM contains three 1024 x 8 bit memories, each is capable of storing the pixel values for one line of data. Lines of data are written sequentially into the three memories. Thus, at any time after the second line following the end of vertical blanking, two of the memories contain the pixel values for the two previous lines, while the current values are being written into the third memory. The pixel counter, which is reset to 0 at horizontal retrace time, provides the address to all three memories. Thus, data being read from the two memories are the values of the pixels directly above the current input pixels (see Figure 21).

A multiplexer connects the memories to a 3 x 3 array of 8-bit registers. This multiplexer is sequenced to transfer the data which was stored two lines before, to the bottom register and data stored one line before to the middle register. Current data is stored in the top register. Also data stored in the first (left) rank of registers is transferred to the second and from the second to the third. Thus, at any time after the second line of a field and the second pixel of a line, the nine registers contain a 3 x 3 array of pixel values.

Nine programmable scalers are connected to the nine registers. The pixel values may be scaled right (divide by 2, 4, 8, etc.), left (multiply by 2, 4, 8, etc.) or do not multiply. The scalers are identified by the circled numbers on Figure 21. The current pixel goes to scaler 7. The center pixel goes to scaler X. Four bits are required to configure each scaler (see Figure 22). Scalers 0, 1, 2 and 3 are configured by the word in address 01, etc. SF0 corresponds to the least significant bit of each group of four bits in the word. If the value is not to be scaled, scale factor 1000 is loaded.

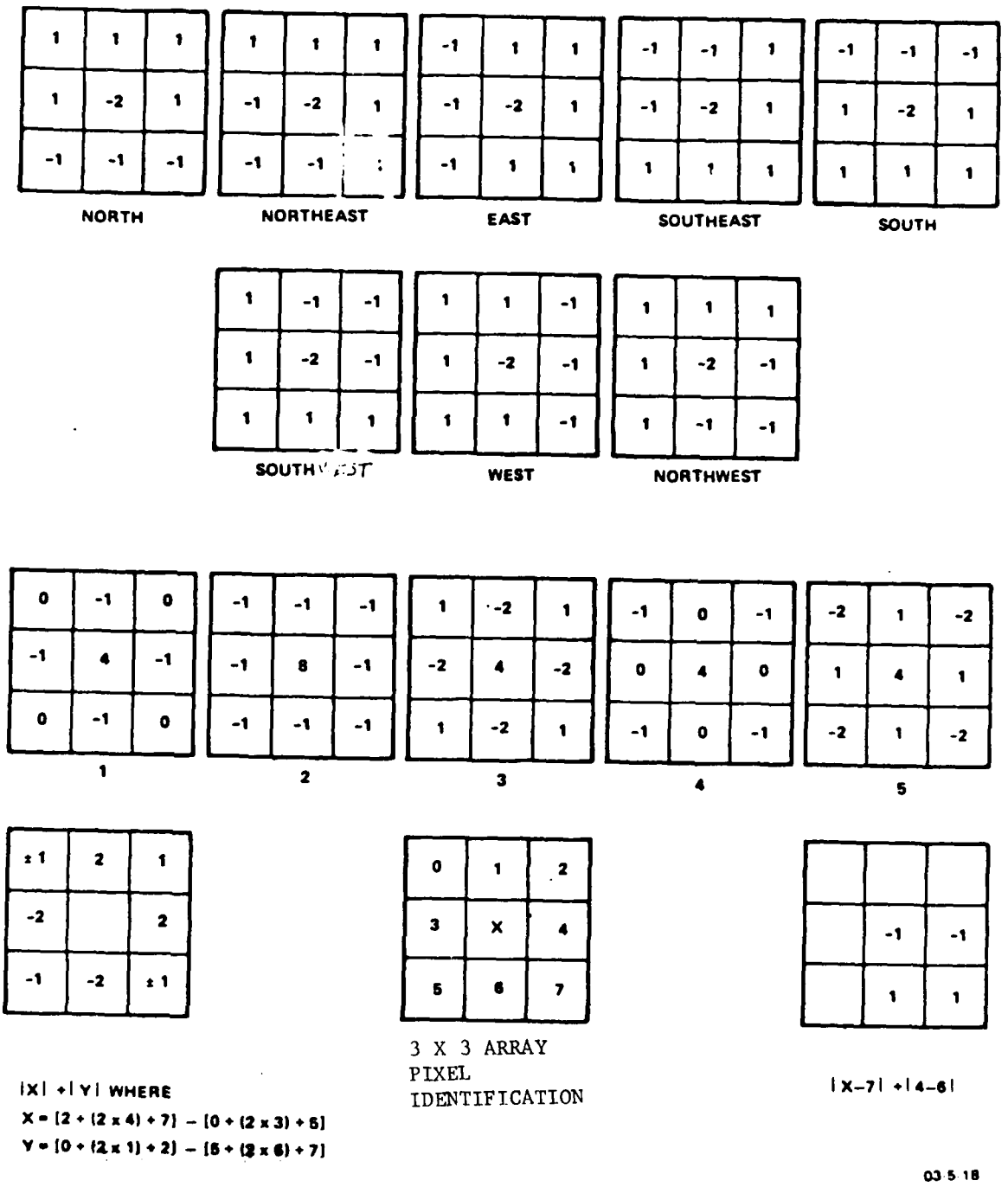


Figure 19. Examples of Video Edge Detection Masks

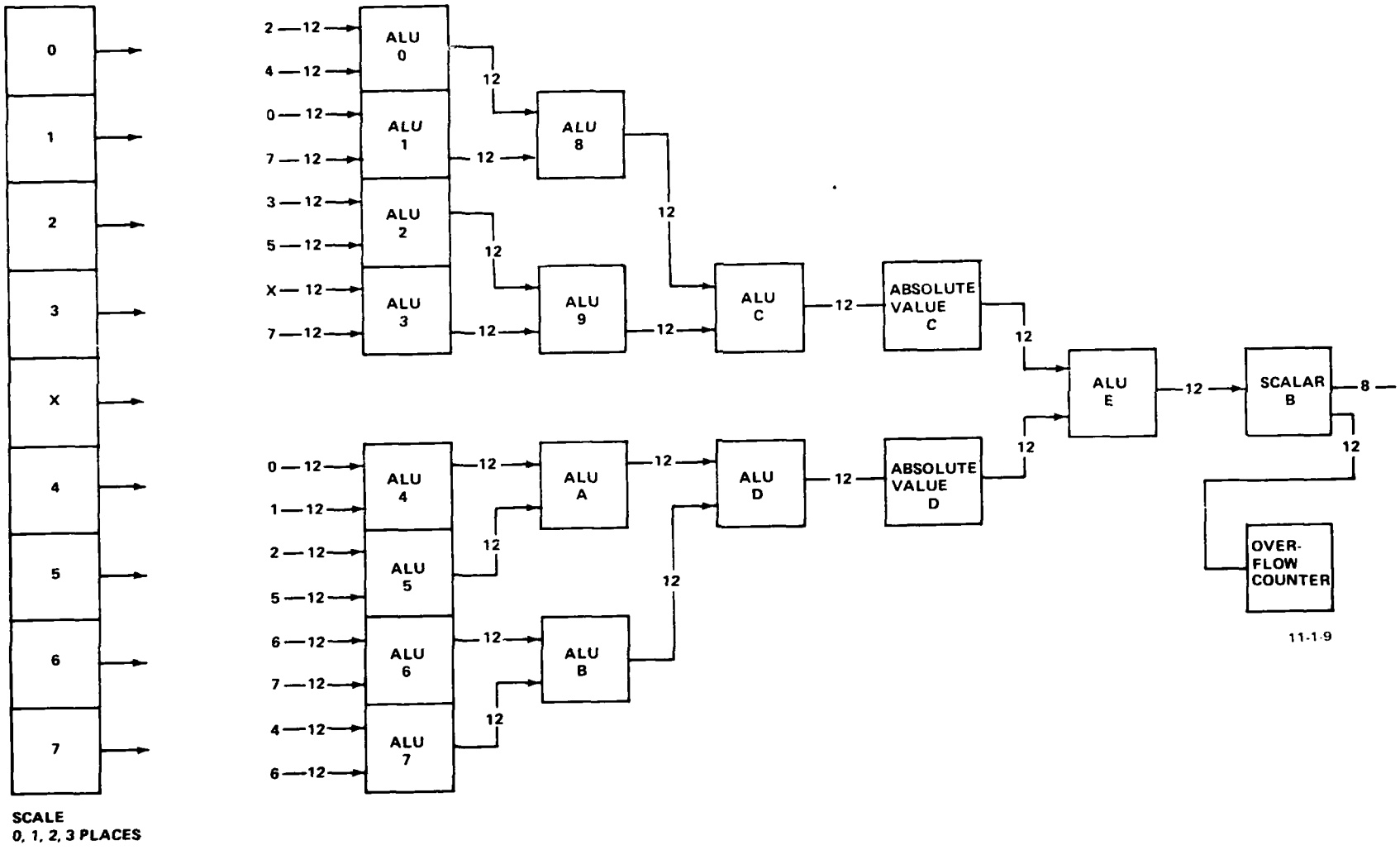
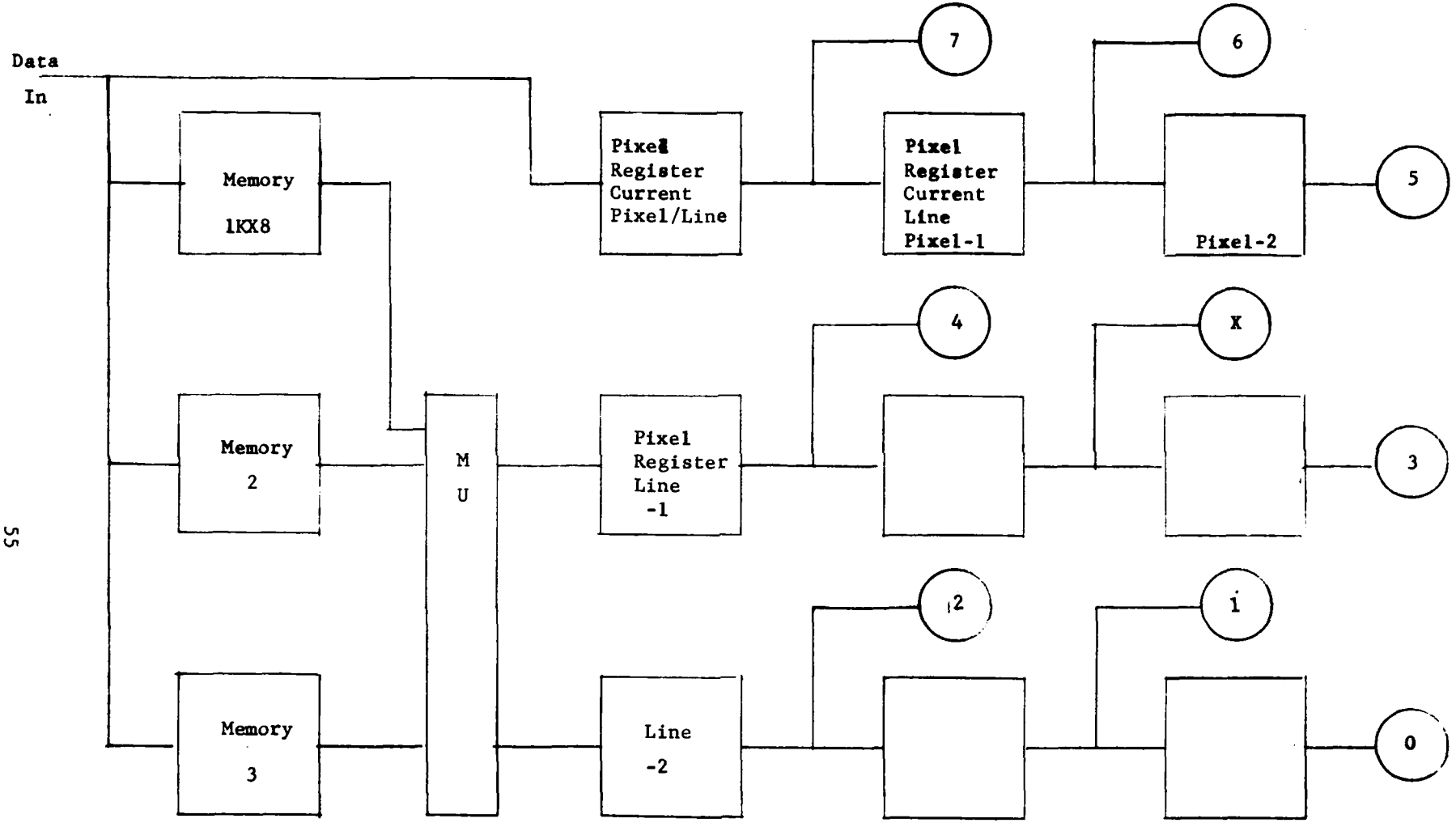


Figure 20. x 3 x mask block diagram



Line	1	2	3	4	5
Write to	1	2	3	1	2
Read from			2 Line 1	3 Line 1	1 Line 1
TO			1 Line 2	2 Line 2	3 Line 2

0	1	2
3	X	4
5	6	7

Figure 21. Video mask

ADDR				DATA													
3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0																
0	0	0	1	Scaler 3				Scaler 2				Scaler 1				Scaler 0			
0	0	1	0	Scaler 6				Scaler 5				Scaler 4				Scaler X			
0	0	1	1									Scaler 8				Scaler 7			
0	1	0	0	ALU 3				ALU 2				ALU 1				ALU 0			
0	1	0	1	ALU 7				ALU 6				ALU 5				ALU 4			
0	1	1	0	ALU B				ALU A				ALU 9				ALU 8			
0	1	1	1	A_{U_D}				A_{U_C}				ALU E				ALU D			
1	0	0	0									OVERFLOW COUNTER							

SCALE FACTOR SELECT

ALU FUNCTION SELECT

56

Scale Factor				Output												
SF3	SF2	SF1	SF0	Ø11	Ø10	Ø09	Ø08	Ø07	Ø06	Ø05	Ø04	Ø03	Ø02	Ø01	Ø00
0	0	0	0	I11	I10	I09	I08	Z	Z	Z	Z	Z	Z	Z	Z
0	0	0	1	I12	I11	I10	I09	I08	Z	Z	Z	Z	Z	Z	Z
0	0	1	0	I13	I12	I11	I10	I09	I08	Z	Z	Z	Z	Z	Z
0	0	1	1	I14	I13	I12	I11	I10	I09	I08	Z	Z	Z	Z	Z
0	1	0	0	I15	I14	I13	I12	I11	I10	I09	I08	Z	Z	Z	Z
0	1	0	1	Z	I15	I14	I13	I12	I11	I10	I09	I08	Z	Z	Z
0	1	1	0	Z	Z	I15	I14	I13	I12	I11	I10	I09	I08	Z	Z
0	1	1	1	Z	Z	Z	I15	I14	I13	I12	I11	I10	I09	I08	Z
1	0	0	0	Z	Z	Z	Z	I15	I14	I13	I12	I11	I10	I09	I08
1	0	0	1	Z	Z	Z	Z	Z	I15	I14	I13	I12	I11	I10	I09
1	0	1	0	Z	Z	Z	Z	Z	Z	I15	I14	I13	I12	I11	I10
1	0	1	1	Z	Z	Z	Z	Z	Z	Z	I15	I14	I13	I12	I11
1	1	0	0	Z	Z	Z	Z	Z	Z	Z	Z	I15	I14	I13	I12
1	1	0	1	Z	Z	Z	Z	Z	Z	Z	Z	Z	I15	I14	I13
1	1	1	0	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	I15	I14
1	1	1	1	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	I15

Scaler

FUNCTION SELECT TABLE

S_3	S_2	S_1	S_0	FUNCTION
L	L	L	L	A plus B BCD
L	L	L	H	A minus B BCD
L	L	H	L	B minus A BCD
L	L	H	H	0 minus B BCD
L	H	L	L	A plus B Binary
L	H	L	H	A minus B Binary
L	H	H	L	B minus A Binary
L	H	H	H	0 minus B Binary
H	L	L	L	F_n $A_n B_n$ $A_n B_n$
H	L	L	H	F_n $A_n \bar{B}_n$ $A_n B_n$
H	L	H	L	F_n $A_n \cdot B_n$
H	L	H	H	F_n A_n
H	H	L	L	F_n $\bar{B}_n$
H	H	L	H	F_n B_n
H	H	H	L	F_n $A_n B_n$
H	H	H	H	F_n LOW

ALU

Figure 22. Video edge detection mask configuration

Eight ALU/Latches are connected to the scalers as shown in Figure 20. The arithmetic functions that may be programmed are shown in the Function Select Table in Figure 22. As with the scalers, each ALU requires 4 bits to program. Configuration word 0100 configures ALUs 0, 1, 2, and 3 and so corresponds to the LSB of each four bit portion of the configuration data word. The ALUs are identified ALUO → ALUE in Figure 21. The A input to each ALU is the lower input, i.e., the A input to ALUO connects to scaler 4 and the B input connects to scaler 2. Also the A input to ALUC comes from ALU 9.

Some masks require absolute values. To accomplish this, two circuits are furnished that will, under configuration control, complement and add 1 to negative numbers. Data bits 12 (output of ALUC) and 13 (output of ALUD) of configuration address 7 (set true) commands the absolute value. This mask may be configured to solve any mask shown in Figure 19.

The output of ALUE may be scaled by the same means as the nine in the front end are.

Two's complement arithmetic with extended sign bit is used throughout this mask. Since scaling of the output of the final ALU is under configuration control and not in hardware, it is possible to have an overflow condition. The resulting value could be very misleading. To prevent this, hardware is included in this mask to examine the output of the output scaler and if overflow exists, output the maximum value - positive or negative - depending on the actual data value. An overflow also decrements an overflow counter that, when it reaches zero, sends a flag (TBD) to the Configuration Controller. Suggested use of this counter is to be preset during vertical retrace time, to the maximum number of overflow that can be allowed. If this number is exceeded, the scaling of the output should be changed.

There are no hardware restrictions on the time of configuration. Configuration requires the Configuration Controller to issue the commands and addresses shown in Figure 22. These data are held in configuration registers until they are changed by new configuration. The exception is the overflow counter which must be reset for each field, if it is to be used.

Data bits 0-2 of address 0000 selects the input source multiplexer for data and blanking. The selection is:

Data Bit 2 1 0	Source Selected
0 0 0	Variable Averaging Mask PP
0 0 1	Sum & Difference
0 1 0	Threshold & Binarization
0 1 1	Double Gate
1 0 0	Peak and Valley
1 0 1	IE Bus
1 1 0	Video Input
1 1 1	ID Bus

Data bits 6 → 9 of address 0000 selects the input source multiplexers for Data Valid or AOI. When Bit 9 is not set, bits 6 → 8 selects AOI inputs 0 → 7 from the PPIM. When bit 9 is set, bits 6 8 selects Data Valid inputs in the same sequence as the data multiplexer above.

Scaling and Arithmetic Configuration (Figure 22) Addresses 1 → 7 are used to configure the Scaling and Arithmetic operations. When Data bits 12 → 13 in Address 7 are set, circuits in the arithmetic pyramid (Figure 21) convert negative numbers to their absolute (positive) value.

Address 8 sets the overflow counter to the contents of data bits 0 → 11.

CIRCUIT CARD DESIGN

The AASAD circuit cards (modules) will be fabricated using the Multilevel/ Multiwire* technology. This method of circuit card construction is closely related to conventional printed circuit technology and significantly superior to the wirewrap fabrication technique. In the areas of weight, high speed electrical characteristics and controlled impedance, multiwire and printed circuit are about equal and both are superior to wirewrapping. For high density packaging multiwire has a slight cost advantage over printed circuit for the same electrical properties.

*Multiwire® is a U.S. registered trademark of Kollmorgen Corporation.

The selection of multiwire in lieu of printed circuit or wirewrap was based on six parameters:

(1) Weight reduction

(2) Smallest package volume

(3) Vibrational acceptability

(4) Electrical characteristics and consistency from card to card

(5) Ease of duplication

(6) Cost

The first four parameters were given the greatest weight in our consideration of these technologies for the AASAD fabrication. The weight and package volume considerations both favor the multiwire and printed circuit due to the pins required on the wirewrap card. The vibrational environment of AASAD would necessitate some method of securing the interconnecting wires to the wirewrap card and even then would be more prone to failure than either printed circuit or multiwire. When considering the electrical characteristics multiwire was a clear "winner" since wirewrap does not provide a constant or controlled impedance and does demonstrate a very high crosstalk characteristic. Multiwires small cross section signal wiring and the right angle crossings provide a lower capacitance coupling than printed circuits where the parallel lines are separated by a few thousandths of an inch and run side by side, over and under other signal lines.

Ease of duplication is clearly in favor of printed circuit and multiwire and cost favors multiwire. Since both printed circuit and wirewrap fabrication techniques are well known, perhaps a brief explanation of multiwire technology is in order.

The multiwire circuit cards are produced in a simple seven step process:

(1) First, the connector plug-in fingers, power and ground plane are defined and then fabricated using a conventional printed circuit board process and epoxy glass laminate.

(2) Next, an adhesive material is applied to the board. It is this material on which the wires are placed. The plug-in fingers and card-guide areas are not covered by adhesive.

(3) Numerical controlled (N/C) wiring machine places the wire pattern on the boards. Each wire begins and ends at a hole location and may intersect any number of hole locations making up the net.

(4) Next the wires are pressed into the adhesive and cured. After the wires have been encapsulated in epoxy, the various component holes are drilled on an N/C machine.

(5) Copper is deposited in the holes by an electroless, additive plating process which bonds each wire end to the wall of the hole.

(6) Fabrication is next. Techniques used here are standard printed circuit manufacturing methods.

(7) Every Multiwire board is given an electrical continuity test then baked dryed and the outer periphery blanked or routed. For better quality control and traceability, lot control numbers are permanently placed on every Multiwire board.

The circuit interconnection uses insulated wires, 34AWG, placed on a standard 100 mil grid (50 mil minimum) with random crossovers as required. Two levels of wires can be placed on each side of the card.

The extensive use of high speed Schottky, ECL 10K and ECL 100K circuits make the electrical parameters of characteristic impedance, backward crosstalk and signal line delay a major concern in the card design. The construction techniques used by Multiwire provide a very close proximity between the signal wire and the cards ground plane, resulting in a more consistent characteristic impedance for the signal line.

The characteristic impedance (Zo) of a single wire over a ground plane is calculated by:

$$Zo = \left(\frac{60}{\sqrt{\epsilon_r}}\right) \ln \frac{4h}{d}$$

where ϵr = dielectric constant of medium separating the wire and ground plane

h = height of wire over ground plane

d = diameter of the wire

For multiwire $\epsilon r \cong 4.5$, the first layer of 34 AWG wire (6.305 mils diameter) is between 9 and 10 mils above the ground plane.

$$Zo = \left(\frac{60}{\sqrt{4.5}}\right) \ln \frac{4(0.01)}{0.0063} \cong 52\Omega$$

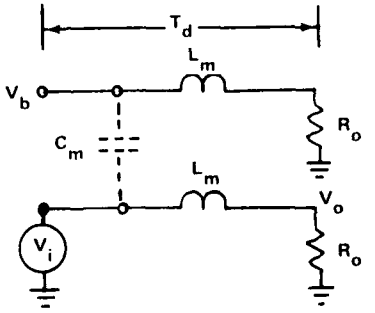
The second level of wiring is 19 to 20 mil above the ground plane and provides a calculated Zo $\cong$ 77Ω. Laboratory tests have measured the first level impedance within a 48-50 ohms range and the second level at about 78 ohms. This impedance is constant over a given card and from card to card of the same type.

The Backward Crosstalk is that portion of the electromagnetic energy coupled from the active signal line to a nearby signal line, via a mutual impedance. This coupled pulse results in a reduction of noise immunity. Two components constitute the total noise, the forward propagating noise wavefront (V_f) and the backward propagating noise wavefront (V_b). The ratio of the peak noise values on multiwire cards is:

$$\frac{V_f}{V_b} = 0.138 \left(\frac{2T}{t_r}\right)$$

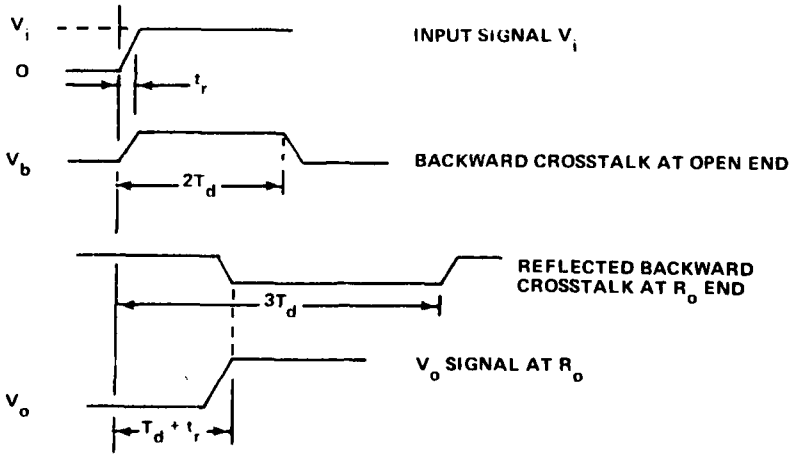
where T is the coupled length expressed in time and t_r is the rise time of the signal applied to the active line.

The open Circuit Crosstalk Model is:



V_i = Input voltage transient to the active line

V_b = The coupled voltage backward noise on the passive line



For $t_r \leq 2T_d$ the backward crosstalk is approximately

$$V_b = K_b V_i$$

For $t_r > 2T_d$ the backward crosstalk is approximately

$$V_b = K_b V_i \left(\frac{2Td}{t_r}\right)$$

where K_b = Crosstalk Coefficient

$$K_b = \frac{}{4T_d}\left(\frac{Lm}{Zo} + CmZo\right)$$

The backward noise increases proportional to the coupled length of the line and is at a maximum when $t_r = 2T\ell$ where $T\ell$ is expressed in units of time. For various permutations and combinations of signal wiring, inter and intralevel active and passive lines and various line lengths the maximum crosstalk noise voltages can be calculated. One typical configuration would be to assume TTL level switching transients of 3 volts/2 ns and a DC low state noise margin of 0.5V the calculations provide a value of $K_b \cong 0.06$ for two active lines adjacent (either side, 50 mils spacing) of a passive line and produce a value of coupled noise (Vb) of approximately 180mV. In general the standard 100 mils spacing will provide a K_b value of 0.013 for the inter layer and 0.032 for the outer layer. When the spacing is reduced to 50 mils these values increase to 0.052 and 0.101 respectively.

These values compare favorably with standard printed circuit technology and are superior to those experienced in wirewrapping construction.

The multiwire cards exhibit a delay time of approximately 1.8 ns/ft which is an acceptable value.

In summary the multiwire fabricated cards provide the AASAD system with a lower weight, smaller volume package and excellent electrical circuit properties.

SECTION 5

NEXT QUARTER ACTIVITY

The planned activity for the next report period, 1 January 1981 to 31 March 1981 primarily revolves around the logic design, mechanical design and the completion of material orders including the circuit cards.

The Digital Video Preprocessors will complete their integration testing in the MVIPS system. Programming and algorithms will be defining additional macros and micros for the program base library. Programming will also start preparation of some program documentation.

APPENDIX

This appendix contains Section 3 of the Technical proposal dated 7 April 1980 and includes editorial corrections found after submission of the Technical proposal and changes made during the MVIPS development. Some of the original contents, that are not required for clarity or technical content, have been removed. corrected/changed areas are identified by a straight line in the right (or left) margin adjacent to the corrected/changed line, figure, table, or formula.

In the section describing the Digital Video Preprocessors there may be features or specifications that are contradictory with the statements or specifications stated in the main body of this Technical report. Where contradictions occur the technical report takes precedence.

SECTION 3

TECHNICAL APPROACH

This section presents a detailed description of FACC approach to the development of the Autonomous Acquisition Simulator System and Associated Data according to the specifications and requirements outlined in RFQ DAAK70-80-C-0416. This description begins with a summary of FACC's proposed baseline design compliance with the technical and derived requirements. The baseline design is then described in detail followed by a description of an optional module called the Integrated Memory Processor (IMP).

3.1 INTRODUCTION AND REQUIREMENTS COMPLIANCE

The compliance of the proposed baseline design to the specification and derived requirements listed in the RFQ are summarized in Tables 3-1 and 3-2. As anticipated, the most severe computation requirements are encountered in the recursive/nonrecursive filtering, image compression and target screening/cueing areas. The recursive and nonrecursive filter implementations require dedicated hardware for filter sizes that exceed minimal dimensions and perform in near realtime. The target screening/cueing algorithms will be implemented by the software programmable high speed digital microprocessor units. Many of the algorithms in these areas are still in a state of development; this means that the AAS system must provide a convenient and flexible procedure for the modification of existing algorithms and the implementation of new algorithms.

In conclusion, the proposed baseline AAS system design will comply with most of the execution time requirements for the typical smart sensor algorithms as listed in section C, Table I of the RFQ. However, with the optional Integrated Memory Processor, the proposed baseline design will meet all of Smart Sensor algorithm simulation requirements.

3.1.1 SUMMARY

In order to meet existing and future requirements, the algorithm simulator design is based upon a clear understanding of the fundamentals of existing smart sensor algorithmic methodologies. As a result of the absence of a definitive all-encompassing solution to the target acquisition, tracking, and identification problems, the baseline AAS design is flexible enough to allow accommodation of the computational requirements of a dynamically evolving field.

It is difficult to anticipate fully the extent and degree to which the present approaches will be modified or supplanted by more successful formulations. However, the progress achieved to date provides a reasonable basis upon which to build quantitative guidelines for an algorithm processing system that allows the user to not only simulate existing capabilities, but also, through a fundamentally software oriented design, to modify and/or add further refinements toward the ultimate goals of the intended weapon systems utilization.

TABLE 3-1. TECHNICAL REQUIREMENTS COMPLIANCE SUMMARY

TABLE 3-1. TECHNICAL REQUIREMENTS COMPLIANCE SUMMARY (Continued)

TABLE 3-1. TECHNICAL REQUIREMENTS COMPLIANCE SUMMARY (Continued)

The next few subsections of the proposal briefly examine the key algorithmic areas of image enhancement, target detection and extraction, target classification, target tracking and bandwidth compression. Only the key concepts and most critical formulations are highlighted with the goal being the derivation of their implicit computational ramifications. A software based system will be able to implement a host of other algorithms that are either modifications of the ones considered here or totally new designs. This is the key asset of a flexible software oriented simulator.

It should be noted that the computational load considerations use the generic term "operation" for adds, subtracts, multiplies, divides and logical comparisons. Although individually distinct, from a computational load point of view, they possess sufficient similarity to be included into one category.

3.1.2 DERIVED SYSTEM REQUIREMENTS

This subsection defines the AAS system requirements from the computational loads imposed by the desired algorithms.

3.1.2.1 Image Sampling (B.3). Image sampling represents the transition from the analog to the digital domain where sophisticated and complex algorithms can be applied to extract information about target and background. Thus it is critical that in this transition one does not lose valuable information and careful consideration be paid to quantization and sampling.

The TV format assumed is 525 line, sampled at a rate of 9.8 MHz. Each horizontal line provides 640 picture elements (pixels). Each pixel will be digitized to 8 bits representing a range of 256 discrete gray levels.

TABLE 3-2. SMART SENSOR ALGORITHM REQUIREMENT COMPLIANCE SUMMARY

For 875 line TV format the sampling rate will be 25 MHz. There will be 720 pixels per line. Based on the Nyquist criterion, the maximum input frequencies will be 4.9 MHz for the 525 line format and 12.5 MHz for the 875 line format.

The internal clock will be synchronized to the incoming composite video on separate vertical/horizontal sync signals. Frame rate of 60 Hertz ±2% will be acceptable. Horizontal and vertical sync will be reconstituted after going through each section of the preprocessors so that frame stabilization will be accomplished.

Sync dropout compensation will include one horizontal line buffer to store the data of the previous line to be used in place of the present line in case of a one sync dropout. In case two to eight consecutive syncs are missing, flags will be sent to the configuration controller to delete the processing of those lines for the present field. In case of nine or more missing syncs, the present field will not be processed.

3.1.2.2 Image Enhancement (C.I). The requirements for image enhancement refer to those aspects of image processing which are intended to accomplish one or more of the following:

(1) Remove "artifacts" introduced by the sensor, readout circuitry, amplifiers and A/D converters.

(2) Enhance, that is increase the gain on, certain image regions, frequencies, or similar parameters of the signal.

In effect, much of image enhancement has the aspects of a bulk-filter, in the sense that the operations to be performed are executed over the entire frame on a repetitive basis. On the other hand, the operations are generally, but not always, of a simple nature.

The AAS will be required to simulate a series of image enhancement algorithms. For the baseline approach, these requirements have been taken to mean "representative" but not "exclusive" algorithms. Image processing algorithms may be naturally subdivided into four all-inclusive categories as follows:

(1) Simple algorithms with modest computational load.

(2) Simple algorithms but requiring severe computational requirements.

(3) Complicated algorithms but modest computational load.

(4) Complicated algorithms and severe computational load.

Most of the current enhancement algorithms fall into classes one and two. A few fall into four, while category three is rarely encountered due to the difficulty in reconciling a complicated algorithm with a modest computational load in a full frame processing environment.

a. Category 1 Algorithms

(1) Detector array compensation

(2) Local area gain brightness control (LAGBC)--simplest algorithm based on a local scene mean and variance.

(3) Simple edge detectors (Sobel, Roberts, etc.)

b. Category 2 Algorithms

(1) Global histogram modification

(2) Convolution type filters with kernels greater than 4 x 4.

(3) Median filtering

(4) Some recursive filters

c. Category 3 Algorithms

(1) Scan jitter elimination through use of a reticle

(2) Auto-focus maintenance

d. Category 4 Algorithms

(1) Local histogram modification

(2) Some recursive filters

(3) Initial focus acquisition

As an example, the LAGBC algorithm of the simplest type is based on the linear transformation

$$V_M = \bar{V}_R + \frac{\beta}{\sigma_R} (V_R - \bar{V}_R) \qquad (1)$$

where V_R is the raw input video, V_M the modified output video, $\bar{V}_R$ the local mean of V_R, σ_R^2 the local variance of V_R and β a gain constant. The quantities $\bar{V}_R$ and σ_R^2 are computed in a recursive manner.

The next level of LAGBC algorithmic complexity, but with similar and actually less severe computational requirements, is represented by the mapping

$$V_M = B(V_R) \tag{2}$$

where B is essentially the inverse function of the global histogram integral.

Further complexity results from operations of the form

$$V_M = B(\bar{V}_R + \frac{\beta}{\sigma_R}(V_R - \bar{V}_R)) \tag{3}$$

which is a combination of the previous two approaches.

Finally, at the highest level

$$V_M = B_L(V_R) \tag{4}$$

where the local function B_L is determined within a window and must be recomputed at each point.

Assuming a 875 x 1024 image size, the number of operations required for each of the above approaches to the LAGBC algorithm are shown in Table 3-3, which illustrates a range of over a decade in computational load.

When other image enhancement algorithms are examined, they are found to fall almost into the same range. Table 3-4 shows the computational loads for diode compensation of staring arrays, push-broom arrays, and median filtering.

The baseline design for the AAS includes the capability for incorporating Category 1 and 2 image enhancement algorithms and portions of Category 4 algorithms into dedicated "preprocessors". The preprocessors will be software controllable modules. One module will hand-off its output to another in a pipeline manner. An example of a preprocessor implementation of equation (1) of Table 3-3 is shown in Figure 3-1.

TABLE 3-3. COMPUTATIONAL LOAD FOR LAGBC ALGORITHMS

LAGBC Algorithm	No. of Operations
(1) $V_M = \bar{V}_R + \frac{\beta}{\sigma_R}(V_R - \bar{V}_R)$	50×10^6
(2) $V_M = B(V_R)$	10×10^6
(3) $V_M = \left(B\,\bar{V}_R + \frac{\beta}{\sigma_R}(V_R - \bar{V}_R)\right)$	60×10^6
(4) $V_M = B_L(V_R)$	600×10^6

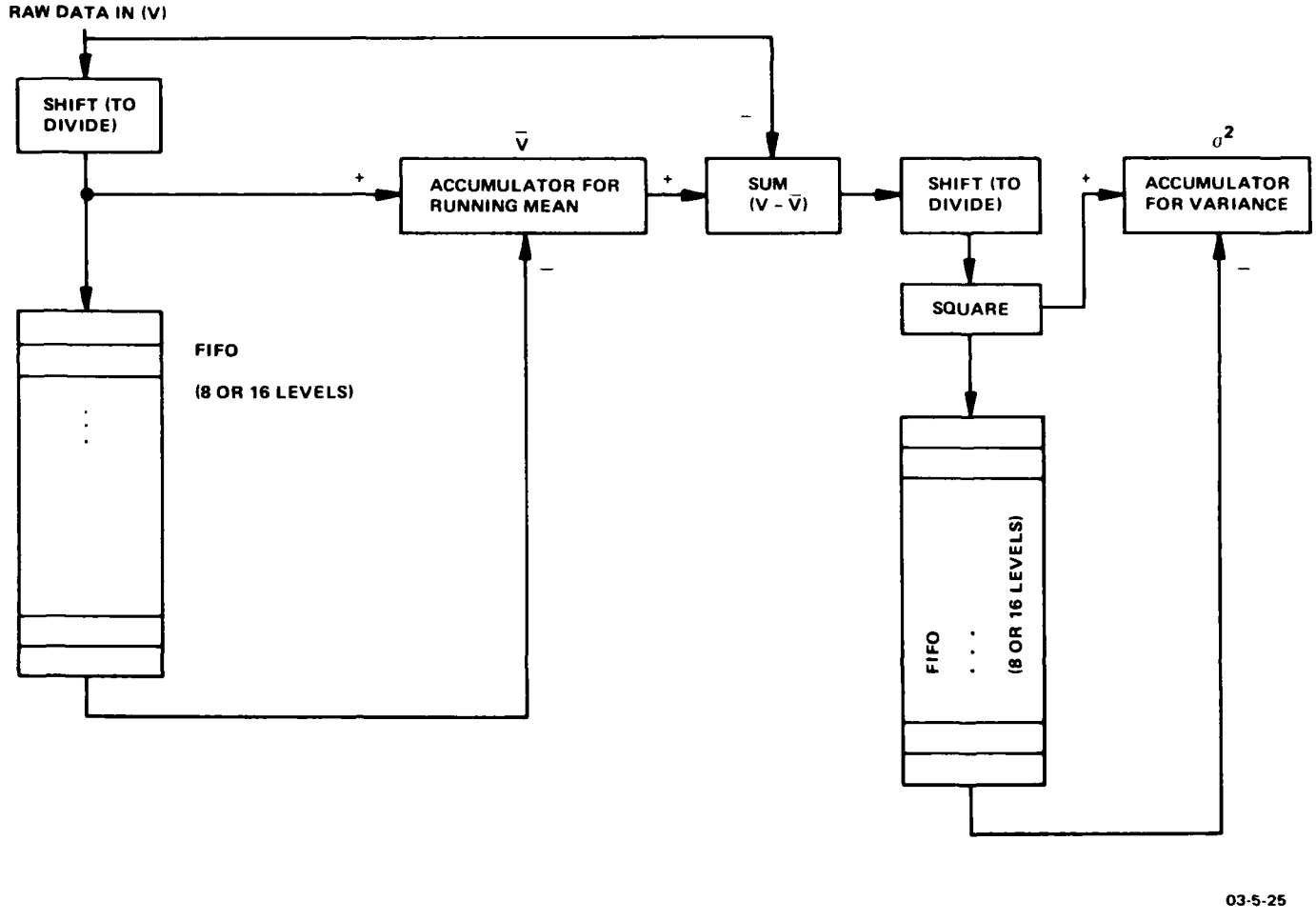


FIGURE 3-1. LAGBC MEAN AND VARIANCE IMPLEMENTATION IN PREPROCESSOR

TABLE 3-4. COMPUTATIONAL LOAD FOR DIODE COMPENSATION AND MEDIAN FILTERING

Algorithm	No. of Operations
Diode Compensation: Staring Arrays	14×10^6
Diode Compensation: Push-broom Arrays	4×10^6
Median Filtering	200×10^6

The image enhancement requirement may be summarized as follows:

(1) Algorithms are required which entail 4×10^6 to over 600×10^6 operations.

(2) The average processing time is about 6 secs. Without dedicated preprocessors such functions may require up to several minutes.

(3) Without some pipelining arrangement when several algorithms are to be chained a problem arises in that

(a) either a large intermediate memory is required (e.g., one frame scratch pad) or

(b) the image must be segmented and the algorithms performed serially on the segments.

3.1.2.3 Target Detection and Extraction (C.II). The ultimate goal of an autoscreening/cueing system is the identification of specific target objects in an arbitrary scene. The current autoscreening/cueing methodologies have characteristically evolved into two distinct phases. One that localizes in the FOV possible target-like objects that possess a reasonable set of fundamental attributes allowing an algorithmically simple prescreening capability against the typically significant number of extraneous clutter scene items. The second phase examines in more detail the individual objects localized in the "preprocessing" phase and is in effect the classifier phase.

There are various approaches to the object localization and prescreening phase. It is imperative to scrutinize most carefully the individual detailed steps of each methodology to ascertain that the algorithm simulator is capable of handling the processing requirements within acceptable bounds. Although at a specific pixel neighborhood the local algorithmic considerations may be fairly reasonable if these are repeated 896,000 times (over a 875 x 1024 image) then the total processing time can be extensive.

Two prescreening methodologies are illustrated next. They are taken from published work (References AD-057-191, AD-060-849 and AD-060-850). The AAS will be able to simulate these and other comparable techniques.

a. Method A (Reference AD-057-191). This technique (outlined schematically in Figure 3-2 and commonly referred to as the Superslice algorithm) first considers a partition of the scene into subframes. Next a median filter is applied for noise smoothing. The median filter replaces each pixel by the median value of the intensity distribution enclosed by a window of predetermined size about the pixel considered. Next a 4 x 4 difference edge detection mask is applied. This mask selects the maximum of the differences between 4 x 4 averages over adjacent pairs of horizontal, vertical and diagonal neighborhoods. This operation is followed by an edge thinning phase considering edge values normal to the direction of a given edge and assigning zero response to the edge of any point within the search mask that has a greater response.

A histogram is constructed for the original smoothed subframe and a set of trial intensity thresholds is selected. The thresholded subframe is then "cleaned" from small and extraneous noise objects by a "shrink/expand" algorithm. This algorithm first eliminates all 1's which have a 0 as their immediate neighbor. Then "expansion" takes place by replacement of a 0 by a 1 if any of its neighbors is a 1. Each phase of the "shrink/expand" algorithm is applied a number of times depending on the size of the object that one desires to eliminate.

A connected component algorithm is next applied that performs the collection of thresholded and "noise" cleaned image points into aggregates corresponding to the various individual disjoint regions in the scene. The basic approach is to raster scan the subframe, labelling topologically connected regions.

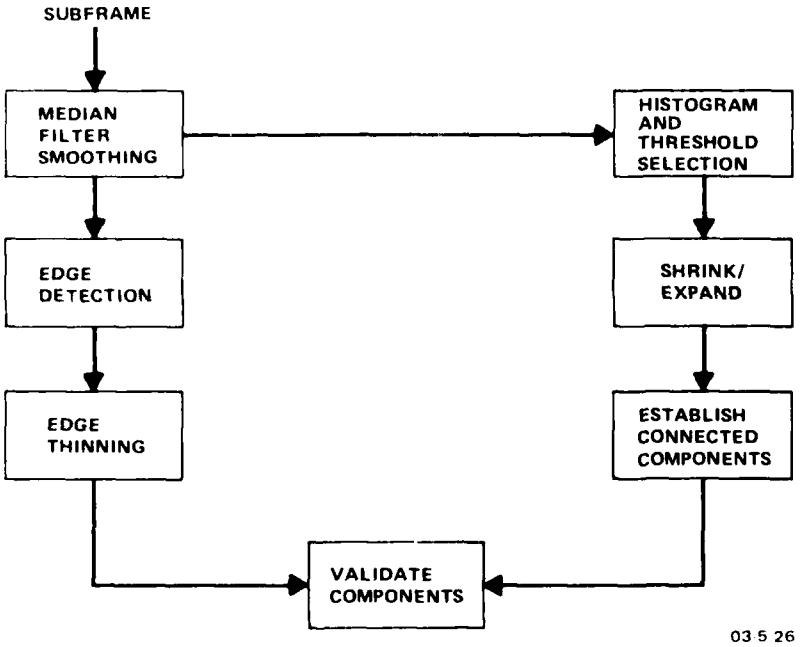


FIGURE 3-2. METHOD A; PRESCREENING ALGORITHM (SUPERSLICE)

Finally, a component validation step is executed which certifies a given connected component as a viable candidate if the following three criteria are satisfied:

(1) Border points coincide with a significant number of edge map points (say, greater than 50% coincidence).

(2) Sufficient contrast exists between border region points and interior points.

(3) Size falls within predetermined range.

For a 875 x 1024 image, the various phases of the above prescreening algorithm lead to the computational load shown in Table 3-5.

b. Method B (References AD-060-849, AD-060-850). This approach has been implemented with analog hardware. In effect it provides an edge threshold and two intensity thresholds (for hot and cold object designation) that allow a raster scan pass over the scene to isolate target-like areas. Schematically the method is as shown in Figure 3-3.

AAS will be able to duplicate with digital computations the full spectrum of this analog processing technique. In particular, it will be able to simulate all the key steps as follows. First an edge map is created through an edge detector of the form

$$\begin{aligned} \text{EDGE} = e = {} & I(n+1, k) + I(n, k) + 2I(n-1, k) \\ & -I(n+1, k-1) - 2I(n, k-1) \\ & -I(n-1, k-1) \end{aligned}$$

TABLE 3-5. COMPUTATIONAL LOAD FOR METHOD A (875 x 1024 IMAGE)

Algorithm	No. of Operations
Median Filter	200×10^6
Edge Detection	32×10^6
Edge Thinning	7×10^6
Threshold Selection	10^6
"Shrink/Expand"	70×10^6
Connected Components	7×10^6
Component Validation	10×10^6

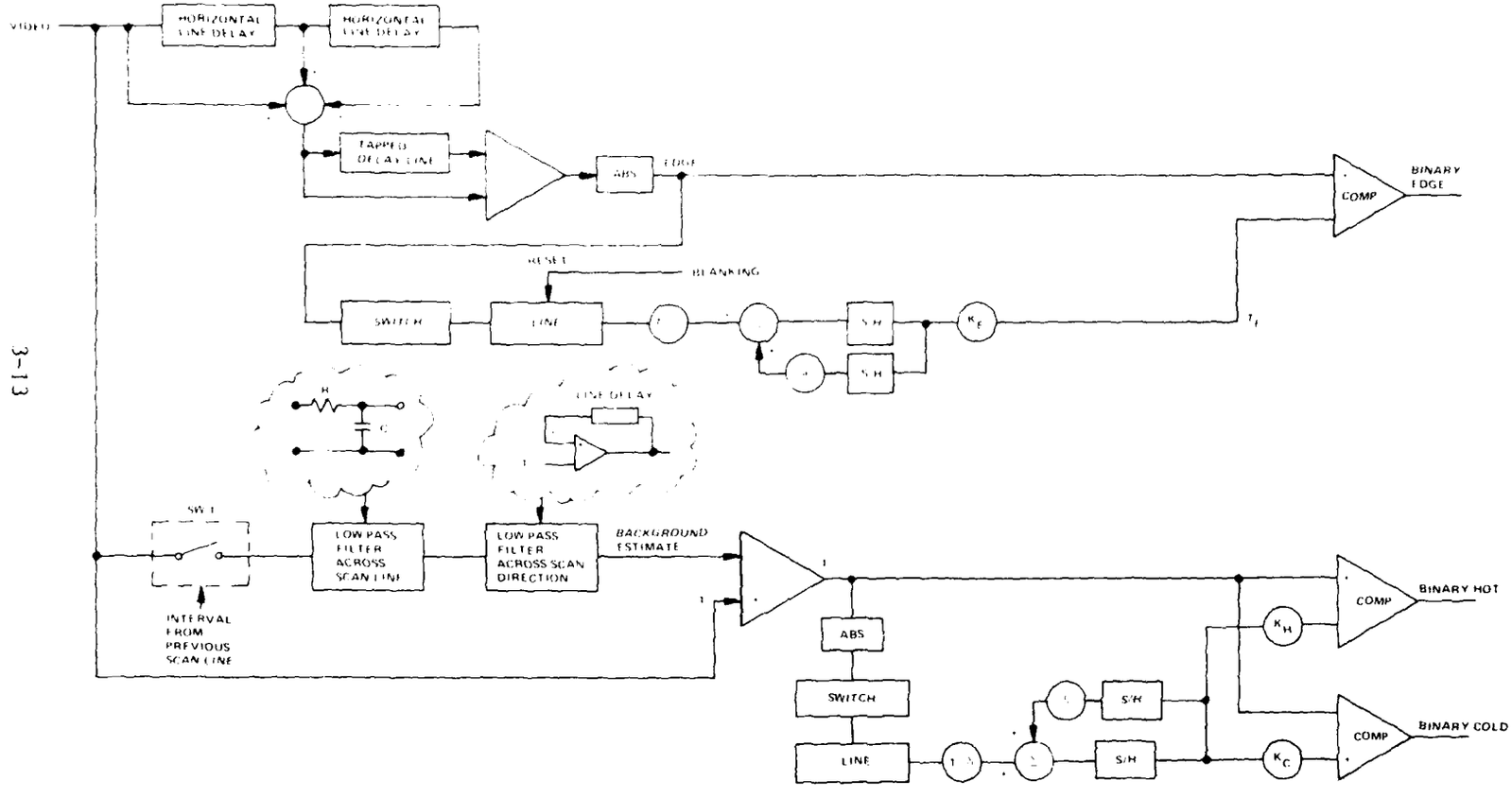


FIGURE 3-3. METHOD B, ANALOG PREPROCESSOR DIAGRAM

where (n, k) is the current row (n) and column (k) pixel under consideration.

Next an edge threshold is computed in the form of a constant times the weighted sum of the last row and next to last row absolute edge values. That is,

$$T_E = K(E) * VAR(N - 1)$$

$$VAR(N - 1) = \alpha * VAR(N - 2) + (1 - \alpha) * \frac{1}{N_R} \Sigma |e|$$

where K(E) is a constant, α is the recursive filter time constant, N_R is the number of row pixels and $|e|$ is the absolute edge value.

A background estimate is also provided by a running background estimator implemented as a low pass filter spanning a given number of horizontal and vertical pixels. After subtraction of the background estimate from the raw scene intensities cold and hot target thresholds are computed through a process similar to the one utilized in the edge threshold determination.

Finally, on the basis of the edge and brightness thresholds target-like object intervals are generated by utilization of three criteria:

(1) Occurrence of an edge at the start of a bright

(2) Occurrence of an edge at the end of a bright

(3) Occurrence of an edge at both the start and end of a bright

Table 3-6 illustrates the AAS computational load for a 875 x 1024 image.

Execution of the full set of algorithms of method A requires approximately 700 x 10^6 operations (with a factor of 2 added to account for the various uncertainties). Method B leads to about 250 x 10^6 operations. With a pessimistic 300 nsec operation execution cycle time AAS will be able to process each method in about 3.5 and 1.5 minutes, respectively.

TABLE 3-6. COMPUTATIONAL LOAD FOR METHOD B (875 x 1024 IMAGE)

Algorithm	No. of Operations
Edge Map	5 x 10^6
Edge Threshold	2 x 10^6
Background Estimate	90 x 10^6
Hot and Cold Target Thresholds	3 x 10^6
Target Interval Generation	3 x 10^6

3.1.2.4 Target Classification (C.II, C.III). After an image has been segmented into targetlike and nontarget regions, the next function of an autonomous cueing system is to actually classify the regions. FACC realizes that in developing a processor for testing such algorithms that the system must be capable of handling a generalized target classifier in which all of the information in a sequence of images is utilized for target identification. This implies that the geometrical relationship between candidate targets as well as the scene structure could be used for determination of each of the target types.

A generalized target classification would function in a manner similar to that shown in Figure 3-4. The pixels for each candidate object (includes clutter) as well as components of a scene are passed to the target classifier from a scene segmentation routine. In principle target classification can be based on a statistical analysis of the target as a whole or for sufficiently well resolved objects it can be based on the geometrical relations of its components. For most systems only one or the other of the two methods would be used to identify an object. But conceivably both methods could be used so that a preliminary target identification would require removal of any conflict between the two object classifiers. A final determination of the classification of each target would be made by an examination of the scene model consisting of any identified scene components and all candidate targets. Their relationship could be based on a few simple scene segmentation rules like "ground target not in sky" or a more formal system could be defined. Any target identification conflicts remaining are resolved (may require subsequent scene analysis) and a final target cueing assignment is made. The scene model has the further advantage of identifying significant structures like a road or a truck convoy to aid the pilot in his navigation.

While is is important that the processor be capable of simulating a complete scene classification scheme it is realized that much of the testing will be done on more basic algorithms where the contextual relations of a scene are either not used or of no interest. Very often it will be necessary to classify the targets before they become sufficiently resolved to even apply syntactical pattern recognition methods. Usually the primary clutter rejection will be based on a statistical scheme for object recognition in order to reduce the amount of computations ultimately required. Consequently it is important that the processor be capable of simulating existing and proposed schemes for statistical target classification.

A statistical object classifier proceeds in two steps. First the essential characteristics (features) of the target are extracted from its image. These features are chosen to represent the qualities of the target which as much as possible uniquely define that object type when compared with other object types of interest. Some measure of similarity is used for a pairwise comparison of the sample feature vector with a set of reference feature vectors. The target category giving the best match (most similar) is assigned to the object being tested.

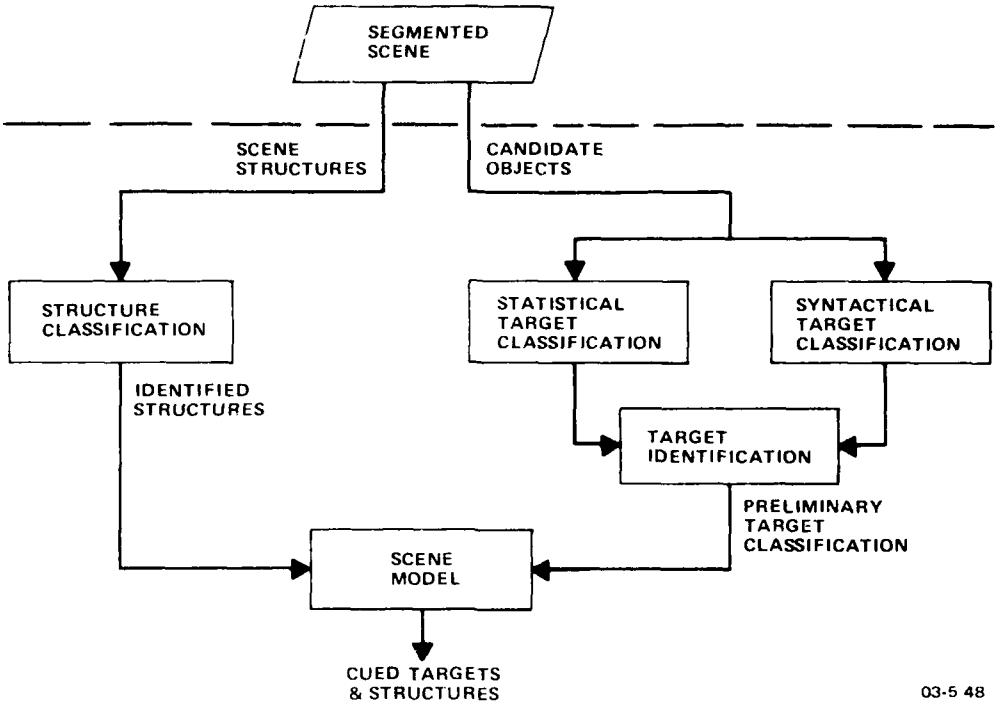


FIGURE 3-4. GENERALIZED SCENE CLASSIFIER

Any target classification scheme to be implemented must be capable of operations in as short a time as possible. Consequently it is common to modify the method just described to reduce the computational burden to manageable proportions. A typical statistical target classifier is shown in Figure 3-5. An initial step has been added to reject as much clutter as possible before the targets are specifically identified. This is implemented using only a two category classification, i.e., target and nontarget. To maintain high throughout only a few simple features are calculated and usually only thresholds testing on the feature values is done. This may be accomplished in a hierarchical fashion using each feature or it may be done for some combination of features such as the Fisher Linear discriminant. A typical set of features might include object area, mean width, mean length, perimeter pixel count, mean peak or minimum pixel intensity and other easily calculable functions based on the object image.

The remaining candidate objects are processed with more computationally intensive features. Ideally a feature vector should be invariant with respect to object size, location, orientation and brightness and should be independent of the order in which the pixels are processed. The feature vector should also be as low dimension as possible yet actually reflect target shape and/or internal structure. Normalized Fourier descriptors of an object's boundary are often used because of their invariance properties. They also have the advantage of being easily calculable using the discrete Fourier transform. Another popular shape descriptor which also considers an object's interior points is the set of seven moments invariant to scale, translation and rotation. Other features are sometimes used which are a function of target size, shape and brightness.

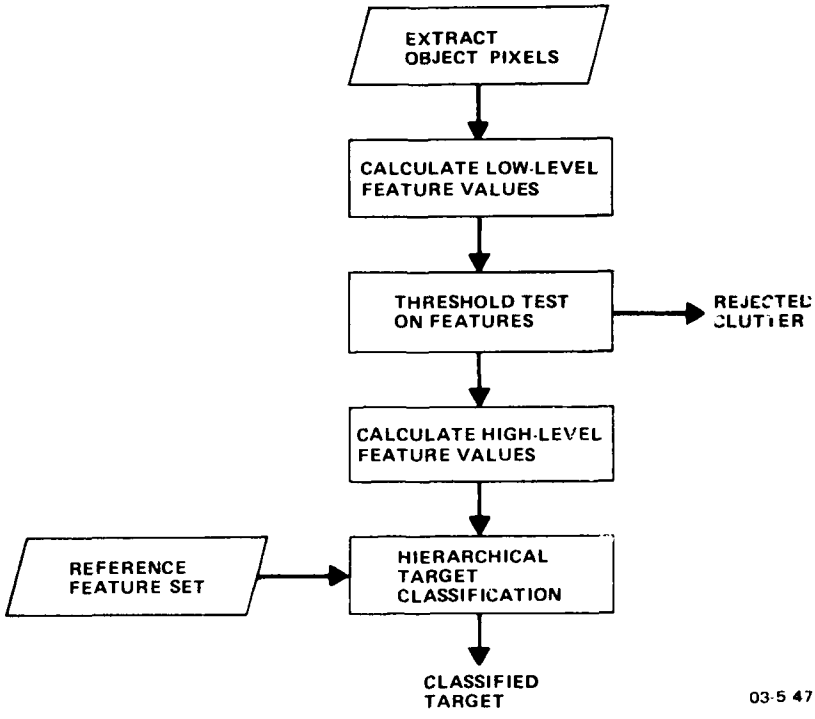


FIGURE 3-5. GENERALIZED STATISTICAL TARGET CLASSIFIER

When a sample (test) feature vector has been calculated for an unknown object it is compared with a reference set of features representing the various possible target categories. It is presumed that for the autonomous acquisition processor the type of military targets of interest, e.g., tank, APC, jeep, etc. are known prior to the mission and the classifier has been trained with images of these targets. The method actually to be used for assigning a target type to an object depends upon how the training data has been categorized and the level of performance required given the constraints of processor throughput and frame time.

One of the most straightforward ways of implementing the target classifier is to utilize the training vector set directly. For example if M images for each of N target types were used to calculate N clusters of M feature vectors, then a K nearest neighbor algorithm can be used directly. The K (a predetermined constant) nearest feature vectors to the sample feature vector is determined. Euclidean distance in the feature space is used as a measure of the separation of the vectors. The object is assigned to the category which contributed the largest number to the set of K vectors.

If the training set is large enough, it may be worthwhile to characterize the classes of features by their statistical distributions. This has several advantages. Decisions can be made based on statistical estimation theory and

a confidence level can be assigned to each decision. A target category or subcategory is specified by a relatively small number of parameters instead of a set of feature vectors. If n features are calculated, the K-nearest neighbor algorithm will require storage of nm values for each target category; whereas if a distribution is used which is specified by a mean feature vector and covariance matrix - a total of n+n (n+1)/2 values must be stored. For example if 10 features are calculated based on 100 images, this would represent a 15 to 1 reduction in storage requirements.

A simple statistical classifier is the Euclidean distance between features normalized by the standard deviation of the feature. This effectively eliminates arbitrary weighting of some features whose values happen to be large compared to other features. The discriminate function for the jth target type would be

$$S_j^2 = \sum_{i=1}^{n} \left(\frac{X_{ij} - X'_{ij}}{\sigma_{ij}} \right)^2 ,$$

where the ' denotes a reference value and X_i is the value of the nth feature. The object is assigned to the category with minimum S^2 value. Confidence in target classification can be improved if the sample feature vector X is replaced by its mean μ as more measurements are made.

The above discriminant can be improved if the correlation between features is taken into account. This implies that the covariance matrix

$$C = E\left[(X' - \mu')(X' - \mu')^T \right]$$

is calculated so that the discriminant function becomes

$$M^2 = (X - \mu')^T C^{-1} (X - \mu')$$

where M is referred to as the Mahalanobis distance. If in fact the features have an n-variate Gaussian distribution

$$f(X) = \frac{1}{(2\pi)^{n/2} |C|^{1/2}} e^{-1/2 M^2}$$

then the Mahalanobis distance is an optimal Bayes classifier and the decision surfaces becomes hyperquadratics. It is this property that makes it so desirable that the individual features have marginal Gaussian distributions. Thus performance can sometimes be further improved if the individual features can be transformed to be Gaussian-like. One such transformation has been used successfully at FACC to match the first four moments of the pretransformed random variable. These transformations (Johnson mapping[TDX-3]) are defined by

$$Z = \gamma + \eta \ln\left(\frac{X-\varepsilon}{\lambda-X}\right)$$

$$Z = \gamma + \eta \sinh^{-1}\left(\frac{X-\varepsilon}{\lambda}\right)$$

$$Z = \gamma + \eta \ln\left(X-\varepsilon\right)$$

where Z is Gaussian distributed with zero mean and unit variance. The choice of which transformation is to be used depends upon the skew and kurtosis of the original random variable X.

For many classification schemes several features are often important in separating classes of objects so that only a fraction of the features set in used to classify most targets. This hierarchical approach may require tests of features or combinations of features but can often reduce the computation time significantly for target classification.

One further capability could be built into the system to optimize the final performance of a classifier. This would involve some scheme for reducing the feature space. The approach could be as simple as merging highly correlated features or if some optimum subset of features is desired, the distance between the clusters for each target type could be maximized for combinations of features. There are many schemes available for selecting the subset of features to be tested, none of which is ideal. However a commonly used measure of the separation of 2 clusters with probability density functions f_1 and f_2 is the Bhattachara distance

$$B = \int_{-\infty}^{\infty} \sqrt{f_1(X)\ f_2(X)}\ dX$$

which for an n-variate Gaussian distribution has an easily calculable formulation.

While it is felt that most target classification algorithms will not require the sophistication of the complete approach just outlined, nevertheless some subset of the algorithms will be used for all classifiers and the processor must be flexible enough to accommodate the complex algorithms required of high performance systems.

Current state of the art classification methodologies are easily handled by the AAS algorithm simulator proposed by FACC. As an example, a preliminary estimate of the throughput required by published techniques (References AD-057-191, AD-060-849, AD-060-850) is as follows.

a. Method A (Reference AD-057-191). In this approach, target classification proceeds in two steps after the "superslice" algorithm has rejected the object-like regions which are nontargets. In the first step (semantic classification), a set of ten candidate features, which can be calculated recursively as pixels are accumulated, is used to preclassify each candidate objects as targets or nontargets. They are easily calculated functions such as object first and second moments, area, intensity first and second moments and so on. If the candidate objects have on the average 200 pixels and if for each feature an average of one add and one multiply per pixel is required then 2000 adds and multiplies would be required to calculate all the features for one object. The number of additional adds and multiplies required to calculate a linear discriminant function is negligible and since the semantic classification involves only two categories, simple thresholding is sufficient (i.e., only two compares per object). Thus for this phase of target classification, the number of adds and the number of multiplies for 500 objects is 10^6 per frame.

The throughput requirements for statistical classification are even less, primarily for two reasons. First there are fewer objects remaining to be classified at this final stage (typically 1/5 of the previous stage or less) and the features are simple functions of the features used in semantic classification. Assume for example that 11 new features are to be calculated, each requiring say 10 adds and 10 multiplies (actually no features require this many calculations) and that on the average two decision nodes are exercised for each target and that a quadratic discriminant function is required at each step. Then if there are 100 objects to be classified only about 50,000 multiplies and 25,000 adds will be required. The total computation burden for the Method A classification approach is on the order of 1.1×10^6 adds and multiplies per frame.

b. Method B (References AD-060-849, AD-060-850). In the Method B approach object classification takes place in 3 stages: clutter rejection classifier, recognition classifier and interframe decision smoothing. Six very simple features are used in the clutter rejection stage where threshold testing is done. Making the same assumptions about object size and feature complexity as was done for the comparable stage of Method A the number of adds and multiplies required to classify 500 objects would be roughly 6×10^5.

The four moments μ_{11}, μ_{02}, μ_2, and μ_{03} are typically used for the recognition classifier stage. This will take less than 1000 adds and 1000 multiplies per feature. Classification is performed by determining the category of the $K=10$ nearest neighbors. If the Euclidian distance to all features of a training set containing say 250 vectors were calculated and then sorted the average number of compares would be less than 10^5 per classified object with

the number of adds and multiplies required for distance calculation being 2000 and 1000 respectively. For 100 objects, the number of compares required for sorting would be on the order of 10^7 and the number of adds and multiplies for all the classification calculations would be less than 10^6 each. We note that if the K nearest neighbor (NN) algorithm is replaced by a fast NN algorithm where the training data is presorted in hierarchical fashion as advocated by Method B then the throughput requirements for the recognition classifier drop considerably.

The final step of interframe analysis of the classified target string involves updating the estimate of the aposteriori probability of correct target identification on a frame by frame basis using Bayes theorem. If for example there are 6 possible target categories and each target is classified say for 10 frames then the total number of adds and multiplies for 100 targets is only on the order of 10^4.

We see that the total throughput requirement for the Method B classification scheme is on the order of 10^7 operations/frame and would probably be much less than this for any fast NN algorithm actually implemented.

3.1.2.5 Nonintelligent Target Tracking (C.II, C.IV). Current target tracking algorithms may be divided into three basic types:

(1) Centroid or hot-spot trackers

(2) Correlation trackers

(3) Feature trackers

A hierarchy chart of trackers is shown in Figure 3-6. Centroid trackers (type 1) are the best developed and have been successfully demonstrated on many programs, including FACC's BCD program.

The type of filtering that any good tracker (centroid or otherwise) should have is updating and filtering of the error signals. By "updating" we mean that we recognize that because of read-outs, calculation time and various propagation delays, the error signal obtained applied to a time which may be as much as 30 msec prior to real time. Velocity filtering can be used to good purpose when a target approaches another bright object.

Correlation tracking (type 2) is often used when the target fills or exceeds the field of view. In addition this type of tracking is inherently more accurate than centroid. Correlation tracking suffers from two problems: (1) it is computationally expensive, and (2) when the object's apparent size is rapidly growing this type of tracking becomes increasingly open-loop since frequent "reference updates" are required. Hence, one encounters:

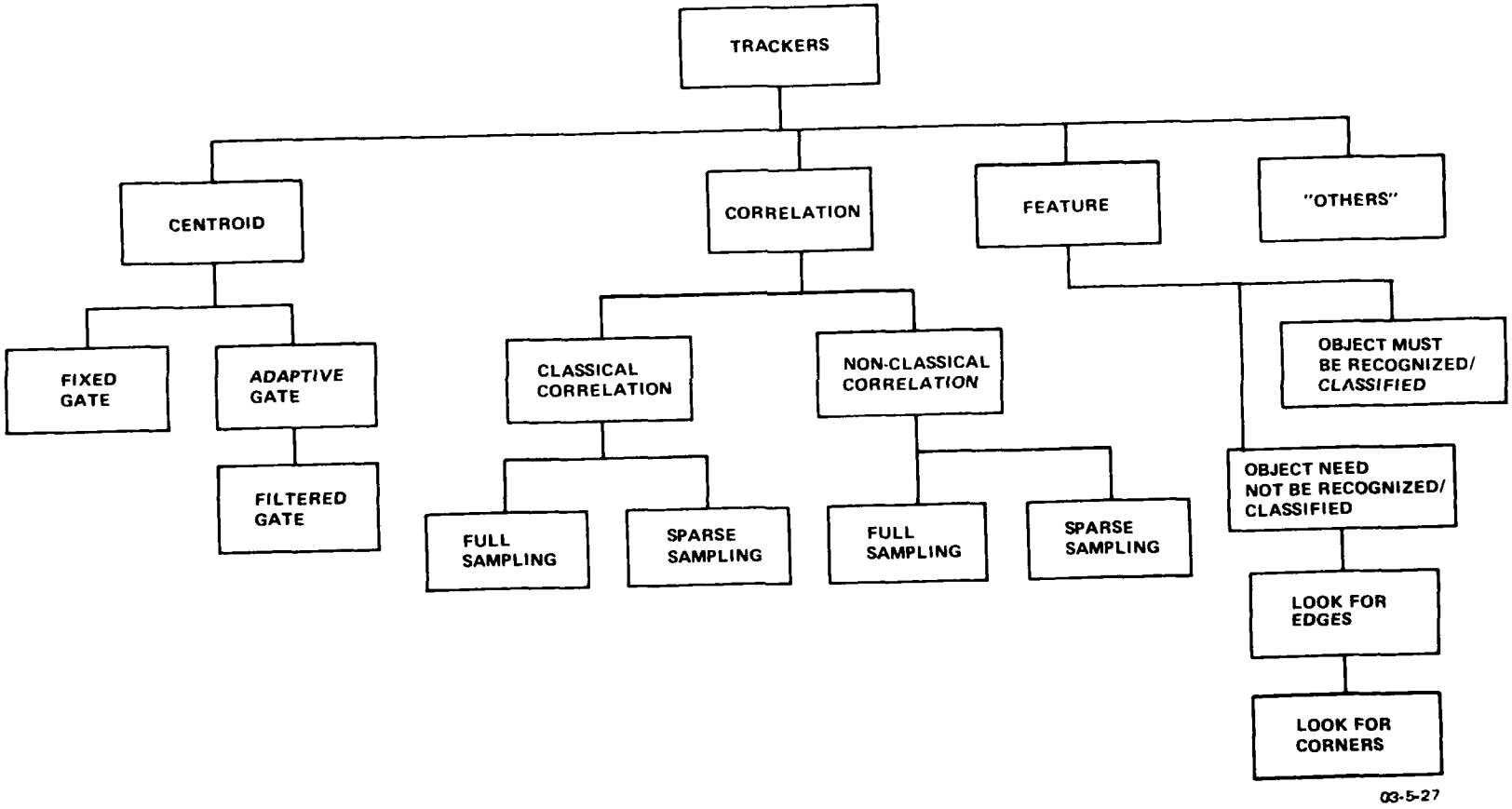


FIGURE 3-6. TRACKER HIERARCHY CHART

(1) Use of maximum gradient techniques to minimize the number of displacements (i.e., correlations) that must be computed to estimate the maximum.

(2) Use of sparse sampling of the target and template to reduce the number of samples that go into a correlation coefficient.

(3) Use of other than classical (least-square) correlation coefficients, for example the L_1 (absolute difference) method.

The third type of tracker is based on recognizing some feature of the target (e.g., fuselage and tail of an airplane, superstructure of ship).

Since the tracker in question must work for a large and unspecified class of objects, such techniques must be restricted to "universal features", i.e., edges and corners.

Multi-mode tracking implies a tracking executive program which is able to adaptively switch from one tracking mode (e.g., centroid) to another mode (e.g., correlation) depending upon background, image growth rate, etc. A throughput of one frame per 2 to 10 seconds is a reasonable upper bound for tracking simulations. Any tracking mode to be evaluated in the field should not operate at a rate slower than 10 seconds per frame, preferably not slower than two seconds per frame. AAS will be able to accommodate these tracking rates in either the correlation or centroid modes.

To get an estimate of the number of operations required we used the following assumptions:

(1) Three objects to be tracked simultaneously.

(2) A gate size around each object not to exceed 64 x 64 pixels.

(3) 100% overhead over basic tracking algorithm to account for (a) tracking executive, (b) track file maintenance, (c) gate and position filtering. Overhead is based on centroid tracking. Same overhead applies to correlation tracking.

(4) In correlation tracking no more than eight correlation coefficients need be computed. (More specifically, four iterations of Fitts' method was assumed.)

The results show that the required operations per frame are

Method	No. of Operations
A. Centroid	0.6×10^6
B. Correlation	2.7×10^6

3.1.2.6 Bandwidth Compression (C.IV). With a view toward reduction of computational complexity associated with a large number of data and size of the memory for storing the data, to bandlimit the data over a channel with a finite bandwidth and to transmit and process the information in real time with a secure and reliable transmission, it is required to derive a scheme to represent only relevant information with the minimum bit per pixel such that extracted information will have a tolerable distortion from the original image of interest. The extraction of only relevant information has been described previously. The amount of bandwidth required to store, transmit and recover the information can be reduced by a) Transform Compression, b) Zonal Compression c) Threshold Compression, and d) Predictive Compression. Zonal Compression and Threshold Compression are part of our algorithms in extracting the information such as target size, shape, scene description and etc.

a. Transform Compression. The inherent correlation of pixels in the original frame causes the energy in the transform domains to be squeezed toward the zero spatial frequencies. It is this characteristic of transformed images that is exploited to achieve bandwidth compression. In addition, the principal components of a feature can be extracted by using Hotelling or Karhunen-Loeve Transform. The following important fast transforms can be implemented to reduce the bandwidth.

1. Fourier Transform. Let $f(m,n)$ be the pixel value located at the (m,n) position of the frame. The two dimensional Discrete Fourier transform is defined by the transform pair:

$$F(k,\ell) = \frac{1}{N} \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} f(m,n) \exp\left[\frac{-2\pi i\ (km+\ell n)}{N}\right]$$

$$, \quad i = \sqrt{-1}$$

$$f(m,n) = \frac{1}{N} \sum_{k=0}^{N-1} \sum_{\ell=0}^{N-1} F(k,\ell) \exp\left[\frac{2\pi i\ (km+\ell n)}{N}\right]$$

where a square image of size N x N has been assumed as the size of the frame.

2. Walsh-Hadamard Transform. The two dimensional Walsh-Hadamard transform of order $N = 2^n$ is defined as follows

$$H = N^{-1/2} H_N,$$

where H_N is a Hadamard matrix of dimension 2^n and is defined in a matrix form

$$F = H[f]H$$

and

$$[f] = HFH$$

where [f] stands for the matrix of picture elements of the frame, F is the transform of [f].

5. Haar Transform. Let H_a be a Haar matrix obtained by sampling the set of Haar functions. The Haar transform pair is given by

$$F(u,v) = N^{-1} H_a [f] H_a$$

$$[f] = N^{-1} H_a F(u,v) H_a$$

Slant Transform. Let S_N denote the N by N slant matrix with $N = 2^n$. The slant transform pair is given by

$$S_N = [f] S_N$$

$$[f] = S_N F(u,v) S_N$$

Cosine Transform. Let transform matrix be written as

$$C = N^{-1/2} [C_{k,\ell}]$$

where $C_{k,\ell} = \begin{cases} 1, \text{ when } \ell = 0,\ K = 0 \\ \sqrt{2}\cos\left[\dfrac{(2K+1)\ell\pi}{2N}\right] & K = 0,1,..N-1, \\ & \ell = 1,2,..N-1 \end{cases}$

The two dimensional discrete cosine transform may be defined as

$$[F(u,v)] = C[f]C'$$

and the inverse transform is given by

$$[f] = C' F(u,v) C$$

where C' is the transpose of C.

6. Number Theoretic Transform: The two dimensional number theoretic transform is written as follows

$$f(u,v) = \sum_{x=0}^{N_1-1} \sum_{y=0}^{N_2-1} f(x,y)\ r_1^{ux}\ r_2^{vy}$$

The inverse transform is given by

$$f(x,y) = N_1^{-1} N_2^{-2} \sum_{u=0}^{N_1-1} \sum_{v=0}^{N_2-1} F(u,v)\, r_1^{-ux}\, r_2^{-vy}$$

where N_1 and N_2 are generally powers of 2 such as 256, 512 or 1024 and

$$r_1^{N_1} \equiv 1 \text{ Mod } M$$

$$r_2^{N_2} \equiv 1 \text{ Mod } M$$

r_1 and r_2 are roots of unity of order N_1 and N_2 respectively.

7. Karhunen-Loeve Transform. Let $R(m,n,p,q)$ be the autocorrelation function of $[f]$, that is

$$R(m,n,p,q) = E\,[f(m,n)\, f(p,q)]$$

(E is the expectation operator)

and $\left[\phi^{u,v} \right]$ be the orthogonal matrix satisfying

$$\sum_{p=0}^{N-1} \sum_{q=0}^{N-1} R(m,n,p,q)\, \phi^{uv}(p,q) = r_{uv}\, \phi^{u,v}(m,n)$$

where r_{uv} are the eigenvalues of the autocorrelation matrix

The KL transform is given by

$$F(u,v) = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} f(m,n)\, \phi^{u,v}(m,n)$$

8. Predictive Compression. If the pixel X_{ij-1} is a certain gray level, then the adjacent pixel X_{ij} along a scan line is likely to have the same value. This can be verified by taking a histogram of $X_{ij-1} - X_{ij}$. Whereas the pixel value range over 256 gray level in the image, most adjacent pixel differences are in a range of about 20 gray level values. Based on the observed value of pixel X_{ij-1}, it is possible to predict the X_{ij}. We suppress the subscript i for the sake of notation. The problem is to estimate X_j given that X_{j-1}. The linear estimater that results in least mean square error $E(X_j - \hat{X}_j)^2$, is given by

$$X_j = \rho X_{j-1} + (1 - \rho) M$$

where M is the mean gray level and ρ is the normalized correlation between the adjacent pixels. Let

$$d_j = X_j - \hat{X}_j, \text{ for all } j$$

The remaining problems are to quantize and code the differences $\{d_j\}$.

1. Differential Pulse Code Modulation (DPCM). The DPCM system uses the previous element to predict the next pixel. The transmitter predictor computes the error between the actual picture value and the predicted value. This error signal is then quantized in a coarse nonuniform manner and transmitted. The received error value is added to the previous predicted value, delayed and possibly attenuated to form the next predicted value in the feedback loop. The quantized output may then be binary coded for error correction and transmitted over the channel.

2. Advanced Intelligent Compression. Algorithms to extract the target size, shape, edge, contour and classification utilize very sophisticated methods which could enable us to compress the data significantly. For example, we used these methods to detect a ship in a background of ocean water, sky, cloud and other associated noise. The ship's location and size are important information, not the cloud or ocean pixel value. The entire frame is of 512 x 512 pixels. Each pixel varies 1 to 255 gray level. The only relevant information is pixel location (row, column) and size. The compression of information is 512 x 512:3. The advanced compression scheme will use image coding techniques to further reduce the bandwidth and enhance the privacy of the message. These schemes include Huffman coding, Run Length coding, Interframe Transform coding, Interframe DPCM coding, Hybrid Interframe coding and BCH coding, and the new technique of image scrambling.

In general, for an image size of N x M pixels, the transform coding requires N^2M^2 operations. However, the fast transforms require $NM/2 \log_2 NM$ operations. The predictive coding requires 2 NM operations. For N = 875, M = 1024, the transform coding requires 8×10^{11} operations and the fast transform requires less than 9×10^6 operations and the predictive coding needs about 2.0×10^6 operations.

3.2 SYSTEM BASELINE DESIGN DESCRIPTION (SOW)

The following subsections present a detail description of the proposed baseline design for the Autonomous Acquisition Simulation system architecture, hardware, and software.

3.2.1 SYSTEM ARCHITECTURE

The system architecture which fulfills the requirements for the AAS system must have a unique blend of hardware and software functions. The requirement of near real time execution of complex image processing algorithms implies the

necessity of fast hardware circuitry with parallel and pipelined processing features capable of executing basic computational tasks germane to image processing techniques such as thresholding,histogram generation and high pass/low pass filtering. The requirement of simulation of diverse image processing algorithms calls for software design that permits the prospective AAS system user to program algorithms in a high level language and that has the capability of translating these algorithms into machine code which can fully exploit the special hardware features especially designed for the simulator. Also the system architecture should be such that it is able to accept any new hardware advances such as special purpose LSI chips or denser memory chips; this will prevent obsolescence in the fast changing world of semiconductor integration. The system architecture should have growth capabilities since it is certainly true that as our understanding of image processing increases, the complexity of algorithms increases proportionally and the proposed simulator should be able to handle them satisfactorily.

A system architecture which satisfies all these requirements is currently not available as "off-the-shelf" equipment. System architectures designed to handle image processing algorithms specifically tend to be specialized in terms of what algorithms they can perform and lack growth features. Almost all general purpose computer systems can simulate the image processing algoritms listed in the RFQ; however they do not either satisfy the size constraints or the timing requirement. Thus we propose a new architecture specifically designed for the simulator which attends to all the requirements called for.

The proposed architecture is presented in Figure 3-7. The system controller handles the interface between the external world and the processing complex. The Control Panel has the provision to accommodate machine code generated for the algorithms to be simulated by a PDP-11 machine in the laboratory. The Control Processor coordinates the computation of the Video Preprocessor and the High Speed Digital Microprocessors (HSDM). The Video Preprocessor (VPP) is specifically designed to handle the analog-to-digital interface (such as sync stripping, digitization and storing frames digitally in the Frame Memory) and simple individual and neighborhood pixel operations. Typical operations it can perform are thresholding/binarization, frame summation/difference and edge operations (e.g., Roberts Mask, Sobel Mask, Laplacian Masks, etc.).

The HSDMs are capable of more general and complex operations. They are designed to handle acquisition algorithms (such as matched filters), extract sub-images, perform correlations and compute image features like histograms and moment invariants. The HSDMs are designed to be independent and execute different instructions at the same time. Also if one decides to replace the HSDM with hardware of his own design, the architecture is designed to facilitate this with its address, control and data bus structures.

The Frame Memory is capable of expansion up to 1 megabytes of storage per frame (four frames maximum). Data and Control Paths are designed so that the HSDMs and VPP can operate independently. The display monitor is used to view raw data, pre-processed data and gate/symbology type of information

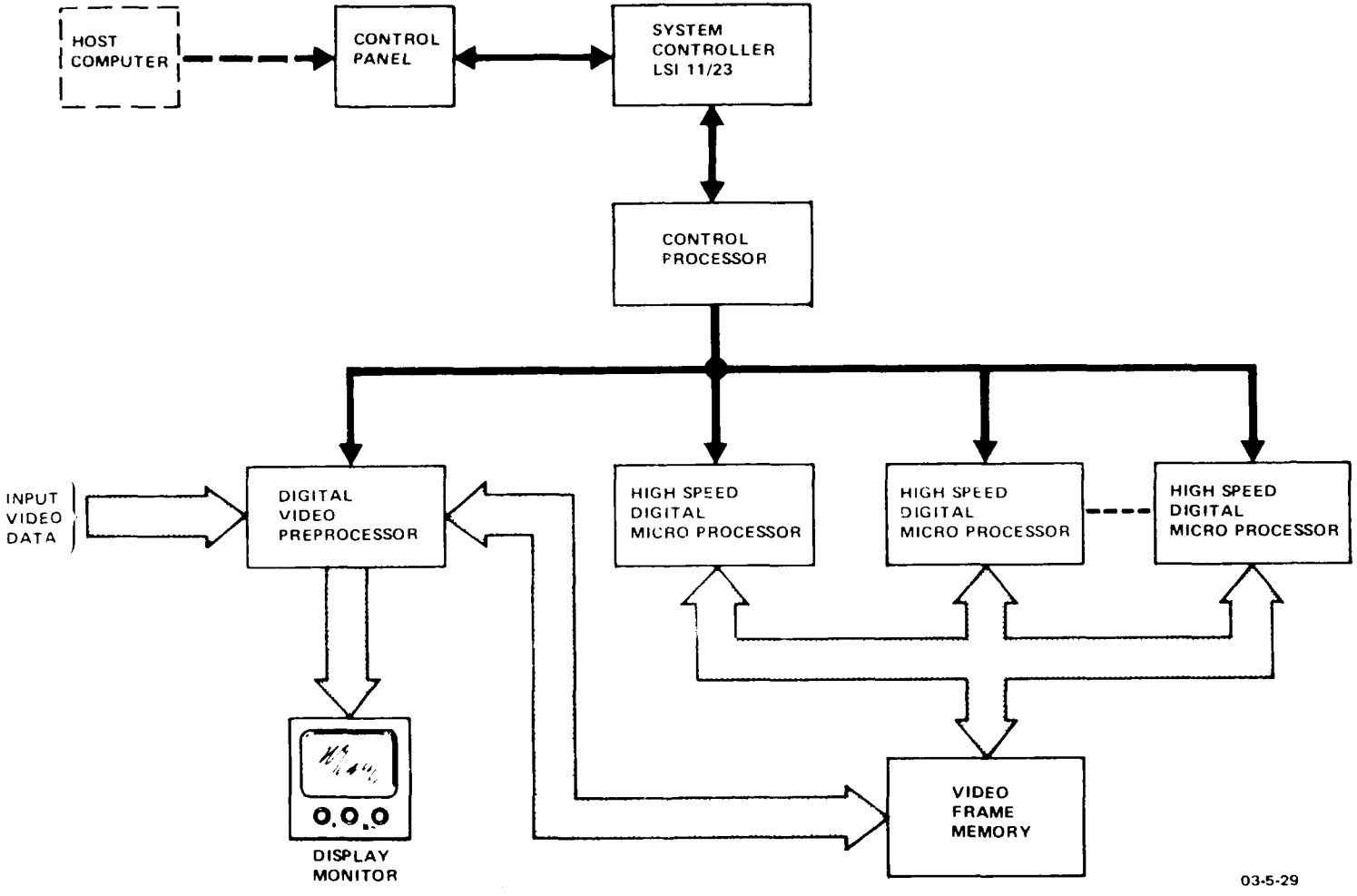


FIGURE 3-7. BASIC SYSTEM ARCHITECTURE

AD-A095 709 FORD AEROSPACE AND COMMUNICATIONS CORP NEWPORT BEACH --ETC F/G 19/5
AUTONOMOUS ACQUISITION SIMULATOR AND ASSOCIATED DATA (AASAD).(U)
JAN 81 S R KING, H MACK, A W MATHE DAAK70-80-C-0235

UNCLASSIFIED NL

2 OF 2
AD
A095709

END
DATE FILMED
3-81
DTIC

The entire processing complex is designed for image processing tasks and hence to exploit fully its hardware capabilities, special attention is paid to macros that implement these tasks on the hardware. The macros are designed to be general and perform tasks such as histogram generation, image correlation, extraction of image subfields, etc. To specify image processing tasks, one uses high level language with subroutine calls and the System Controller converts these calls into macros and pointers to subroutine arguments.

Each individual processing element of the system is discussed more fully in the following subsections.

3.2.2 SYSTEM HARDWARE DESIGN (SOW)

This subsection contains a detailed description and statement of work for the baseline design on the proposed Autonomous Acquisition Simulator (AAS) System processing hardware. This baseline has been derived primarily from the results of a current and on-going internally funded program at the Aeronutronic and Western Development Laboratory (WDL) Divisions of Ford Aerospace and Communications Corporation.

The internal program at the Aeronutronic Division is formally known as the Digital Imaging Search and Track Independent Research and Development Program. This is a multiyear program whose major objective is the design and development of a processor called the Modular Video Image Processing System (MVIPS) and whose basic design meets or exceeds the current requirements for the AAS system. As currently conceived, the MVIPS design includes the baseline design for the Digital Video Preprocessor (DVPP), the High Speed Digital Microprocessor (HSDM), the digital Frame Memory, and Associated System Interface modules. Figures 3-8 and 3-9 are photographs of the MVIPS hardware excluding the DVPP which is currently undergoing system tests and integration in the Digital Laboratory at Aeronutronics.

The Digital Video Preprocessor is currently in the final design stages and it is anticipated that this module will be fully tested and integrated into the MVIPS prior to the initiation of the AAS program.

Finally, an optional addition to the baseline design, called the Integrated Memory Processor is currently fully operational at FACC's WDL Division Laboratory. It should be noted that as currently designed, the IMP meets or exceeds the AAS requirements for two-dimensional filtering, convolution, correlation, etc., in almost real time.

Because of the maturity of the designs for existing hardware and the fact that this hardware meets most AAS program requirements, the approach for the hardware design for the AAS program is to use the existing MVIPS design as the baseline design for the AAS program and to anticipate minor modifications of

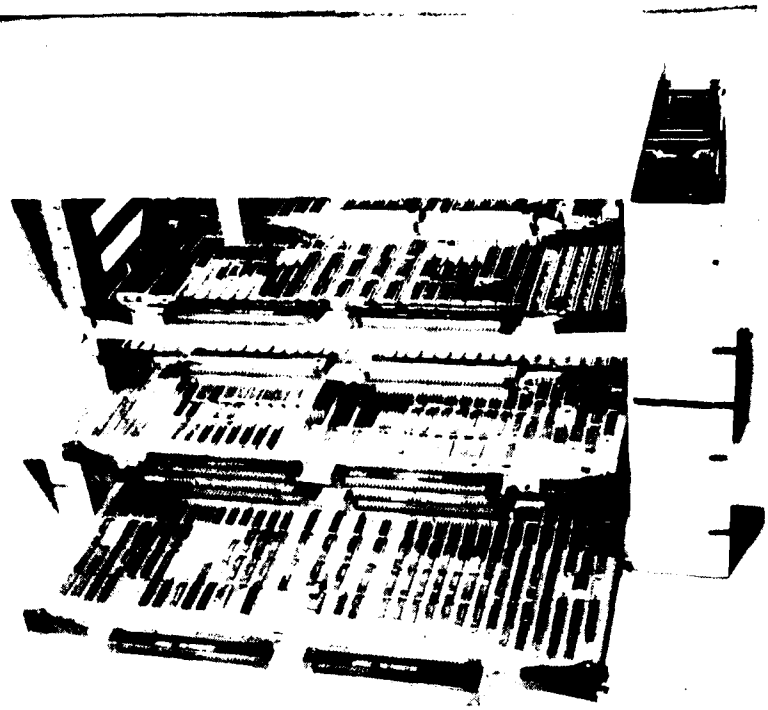


Figure 3-8. MVIPS - Demonstration system

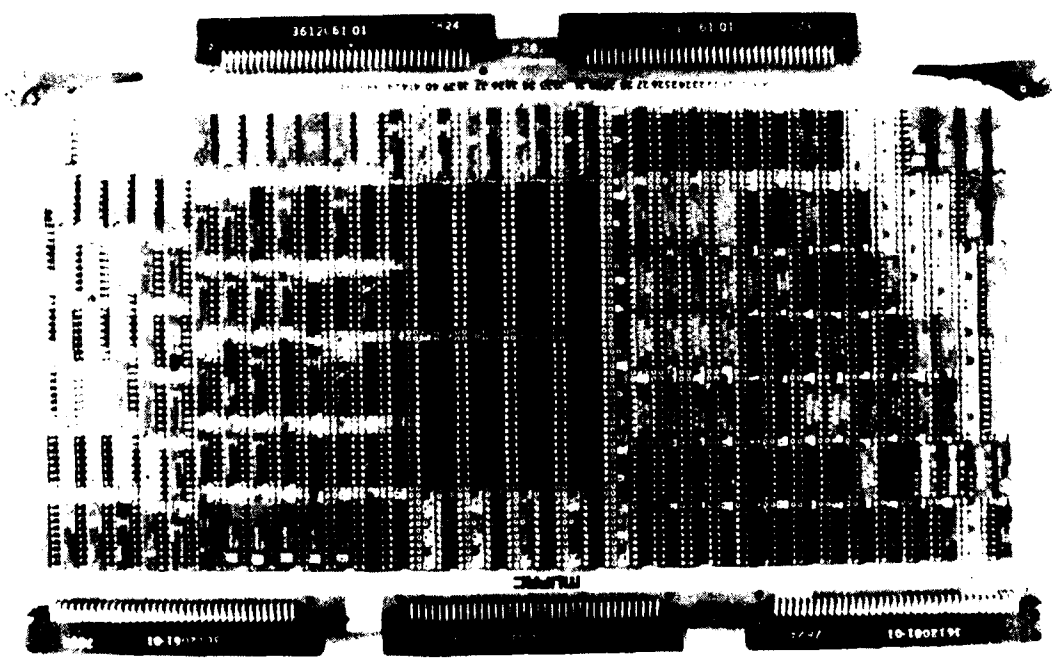


Figure 3-9. MVIPS Sequencer and address generator

this design as a result of the system architecture review task during the very early stages of the program. Moreover, this approach is considered to be one of low technical risk while meeting the AAS program objectives. Of course, there will be some new electrical and mechanical design such as that associated with the control panel, laboratory program load module, and packaging to meet the helicopter environment. However, these tasks are considered low technical risks because of their corresponding straightforward designs.

The following subsections outline the details of the baseline designs for the AAS hardware.

3.2.2.1 AAS System Design Summary. The AAS is designed to be a modular state-of-the-art, high speed realtime image processing system. Maximum use of parallel and pipeline architecture is employed to achieve the processing speeds required for nonrealtime system such as AAS. The design is modular in the sense that additional processors may be added in a parallel lockstep instruction - SIMD (Single Instruction Multiple Data stream) mode or in a parallel asynchronous instruction - MIMD (Multiple Instruction Multiple Data stream) mode to meet the requirements for realtime processing of the AAS algorithms. Furthermore, the requirements for these additional processors have been anticipated in the baseline design and as a result these processors may be added in the future without major modifications to the then existing hardware.

The baseline design for the AAS hardware accommodates the requirement for a High Level Language (HLL) by incorporating a commercially available processor, the Digital Equipment Corporation LSI 11/23 that already has a very large software support base. This support includes many HLL's such as Pascal which has been selected as the HLL for the AAS program.

The AAS hardware is designed to contain a total of nine functional subsystems as shown in Figure 3-10 and are as follows:

(1) System Controller

(2) High Speed Digital Microprocessor

(3) Digital Video Preprocessor

(4) Frame Memory

(5) Graphics Display Controller

(6) Control Panel

(7) Transportable Memory Loader and Fixture (Lab Program Loader)

(8) Integrated Memory Processor (Optional, section 3.3)

(9) Analog Video Processor (interface provision only)

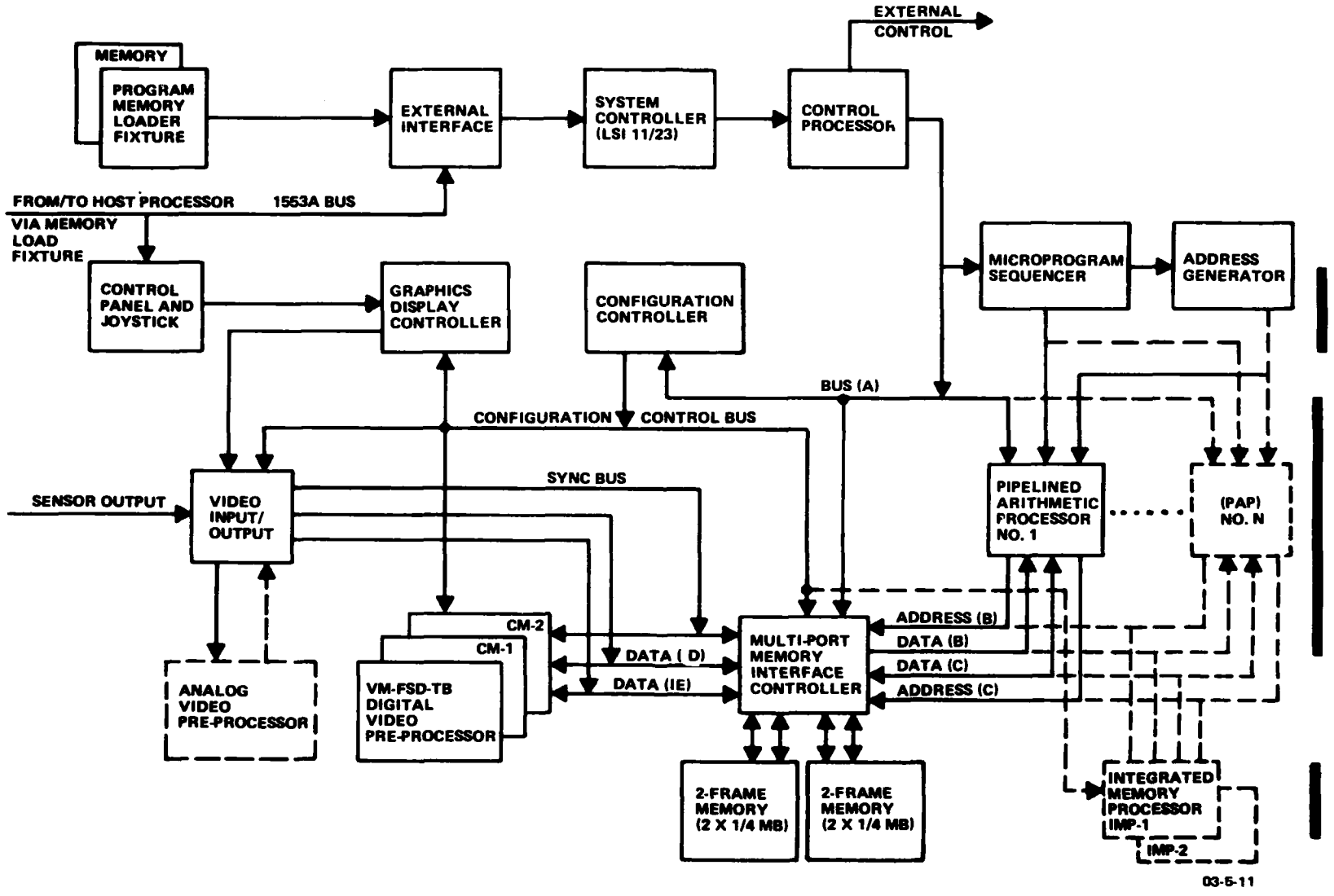


FIGURE 3-10. AAS PROCESSOR BLOCK DIAGRAM

It should be noted that the baseline designs for the AAS program will include the System Controller, High Speed Digital Microprocessor, Digital Video Preprocessor, Frame Memory, Control Panel, Transportable Memory Loader, and Graphics Display Controller and associated system interface hardware. The Integrated Memory Processor is included as an option only. Analog Video Preprocessor is not included in the baseline design or as an option, however, the baseline design does include provisions for interfacing an analog processor to replace or operate in parallel with the Digital Video Preprocessor.

The designs for these functional subsystems are described in detail in the following subsections.

3.2.2.2 System Controller (B.4). The System Controller subsystem will function as the basic controller and computer for the proposed AAS. Its principle functions will be as follows:

(1) Provide for high level control of AAS,

(2) control architecture of AAS,

(3) provide firmware or read only memory (ROM) for the AAS operating system,

(4) provide Random Access Memory RAM) for application program,

(5) provide interface support (hardware and software) for external control signals, and

(6) provide the AAS with a basic computational capability, e.g., floating point operations.

The Digital Equipment Corporation (DEC) LSI 11/23 Computer has been selected as the processor to perform the System Controller function. This processor has been selected because it is software compatible with NV/EOL DEC PDP 11/70 as well as the DEC VAX 11/780 which is currently being acquired by Aeronutronic. This will insure that AAS program has adequate software development support at both the contractor's and customer's facilities without the purchase of additional equipment. This selection also allows for advantage to be taken of the widely available software (e.g., Operating System, Compilers, and Utilities) for the DEC LSI 11 in order to reduce the cost and technical risk associated with potentially very costly software development tasks.

The selected LSI 11/23 has the following features which will be used to meet the requirements of the AAS programs:

(1) DEC PDP 11 family software compatibility

(2) Small size (Four 5.2 x 8.9 in multilayer boards)

(3) Memory management, i.e., software controllable extended addressing, and memory protection

(4) Floating point arithmetic instructions

(5) Vectored interrupts

(6) 8K Bytes of ROM memory

(7) 48K words (16-bits) of RAM memory, optionally expandable

The organization of the LSI 11/23 or the System Controller memory is shown in Figure 3-11. The ROM memory section is basically the system firmware and is designed to contain the DEC LSI 11 Operating System, subset of the DEC utilities program, the Pascal P-Code interpreter, the AAS Executive Software, and the AAS utilities and diagnostic software. This software will be loaded in PROM and will be delivered with the AAS system. The random access portion of the System Controller's memory is designed to contain the user application code. The System Controller is designed to accept application programs from the Transportable Memory Loader as shown in Figure 3-10. The baseline system will contain 8K Bytes of ROM and 48K words of RAM.

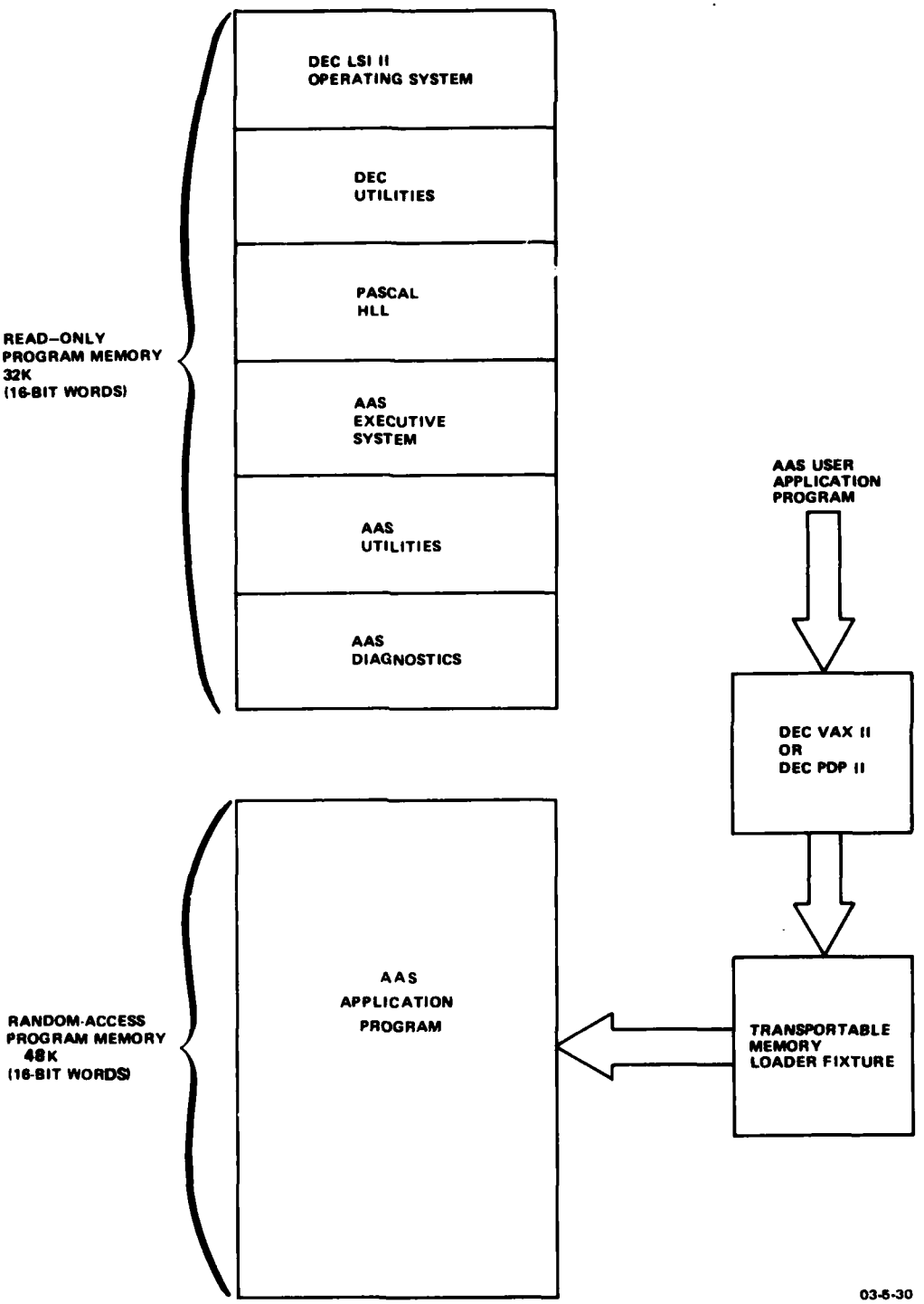


FIGURE 3-11. SYSTEM CONTROLLER MEMORY ORGANIZATION

3.2.2.3 Digital Video Preprocessor (DVPP) (B2) (B3) (B4) (B5). The Digital Video Preprocessor is included in the baseline design for the AAS; its major functions are video input/output processing and high speed video preprocessing operations. The video input/output processing includes the analog-to-digital conversion of 525 and 875-line tv format data, video sync stripping, line drop-out compensation, and digital-to-analog conversion of digitized video data for output to 525 or 875-line tv video monitor. The video preprocessing functions for the DVPP consist mainly of those image processing operations where typically the same arithmetic operation has to be performed on every pixel within a video frame and in realtime. Included in this DVPP design are the operations that would normally be performed by front end analog processing in the more conventional architectures for autocueing/screening processing. Furthermore, the digital approach to the preprocessing has the advantage that it is very flexible and easily expandable. This will enable the proposed AAS to meet future algorithm simulation requirements with minimum modification to the baseline system. Finally, many of the digital implementations of pre-processing lend themselves readily to VLSI design, and therefore, advantage can be taken of this rapidly evolving technology in the near future to reduce the size, weight, and power requirements by orders of magnitudes.

In its current configuration, the DVPP hardware is arranged as a group of special function modules that digitize the video data and perform the arithmetic operations on the data as commanded by the configuration controller. Each special function module is contained on one or more wire wrap circuit boards, 7" x 15" in size. The modules are interconnected and connected to the memory by control and data buss's contained on the back panel. Modularity in design and spare card slots in the chassis provide for expansion or change of function by a simple addition or exchange of circuit cards in the chassis.

These modules are defined as:

• Video I/O	2 cards
• Configuration Control	1 card
• Variable Matrix	1 card
• Threshold and Binarization • Frame Sum/Difference	1 card
• Video Mask	1 card

a. Overall System Functional Organization. The overall functional organization of the Digital Video Preprocessor is shown in Figure 3-12. The principle sub-functional elements of the DVPP are the video input/output, system configuration control, and modular video preprocessor functions. The DVPP system is essentially designed to accept either 525- or 875-line tv format analog video data as defined by the configuration control processor. The sync signals, namely the horizontal (line) and vertical retrace (field), are stripped by

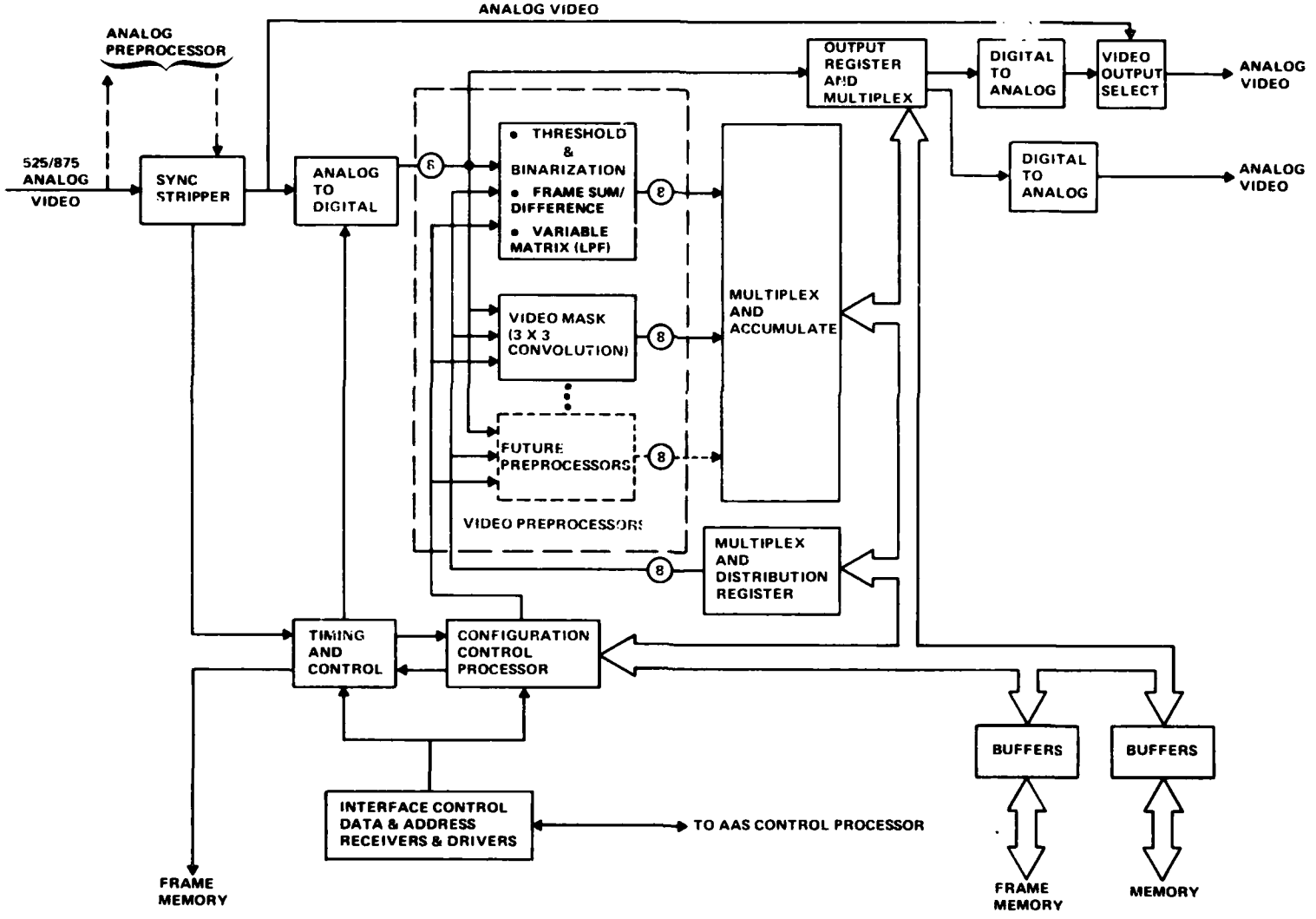


FIGURE 3-12. DIGITAL VIDEO PREPROCESSOR

the timing and control section to generate timing and control signals for the AAS system. Internally, the DVPP digitizes the analog video and processes the data in realtime using dedicated software selectable and controllable preprocessor modules. The system is designed to output two separate processed or unprocessed video data streams in analog form to a 525- or 875-line tv monitor or in digital form to the AAS frame memory. Capability is also provided to read data from the frame memory and process the data again (at video rates) with the preprocessors to facilitate recursive operations. 875-line processing requires that frame memories be configured as 1 megabyte memories.

Finally, the DVPP is designed to output unprocessed analog video to an external device and accept the data from the external device to allow the use of analog processors in a cascaded or parallel mode with digital preprocessor modules, i.e., the data from an analog processor may be sent directly to the DVPP analog video section, or the data may be digitized, optionally processed by a selected preprocessor and then sent to any combinations of the analog video outputs, frame memory, or fedback to a preprocessor.

The following paragraphs describe the subfunctional elements of the DVPP in regards to their function in the system.

b. Video Input/Output Subsection (B2) (B3) (B5). The detailed functional block diagram for the DVPP Video Input/Output subsection is shown in Figure 3-13. This design provides for the following features:

- Input 525- or 875-line tv format data from any source, e.g., FLIR, tv camera, or video tape recorder
- Controls gain of input video amplifier to increase dynamic range
- Inputs for external video sync signal
- Outputs data to an external device for parallel or serial processing with DVPP preprocessor function
- Inputs analog data from an external analog processor
- Generates AAS sync and blank signals from input video or external sync signals
- Digitizes analog video up to a 25 MHz rate at 8-bits per sample
- Outputs digitized data to the DVPP preprocessors and the internal ID and IE buss's.
- Outputs 525- or 875-line analog video up to two tv monitors
- Output Analog video is selectable from output of A-to-D converter, input of A-to-D converter, and the DVPP internal ID and IE buss's.
- Mixes character and graphics data with analog video

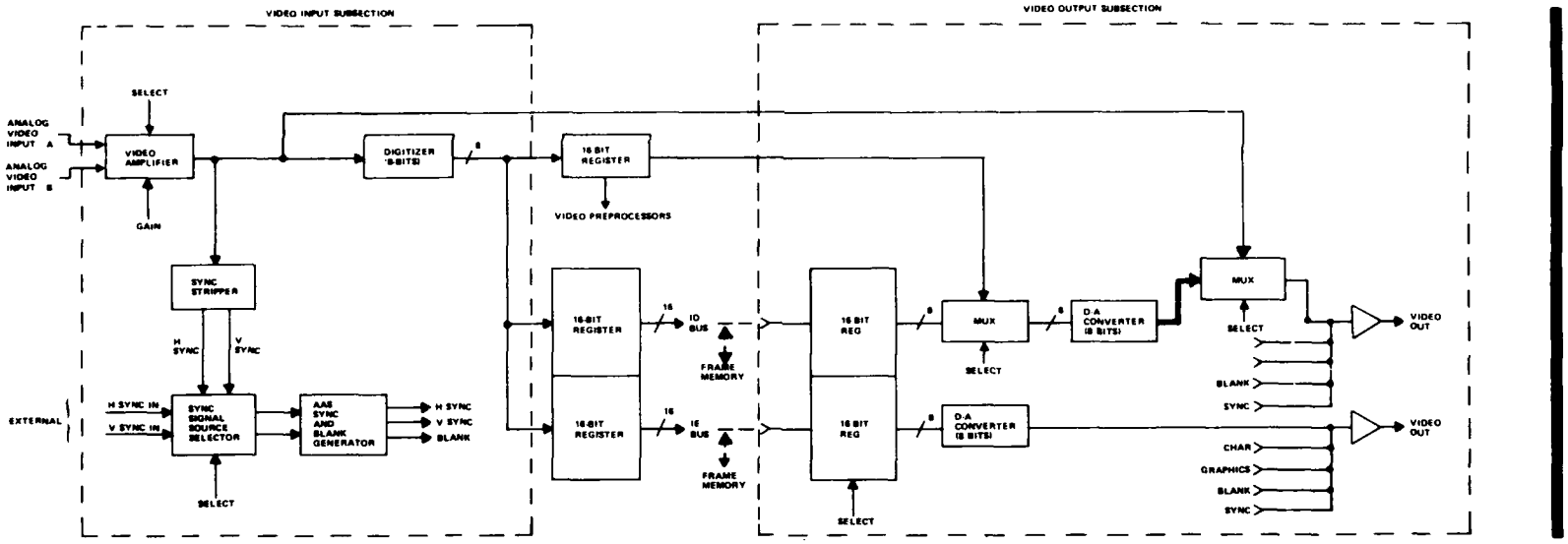


FIGURE 3-13. DVPP VIDEO INPUT/OUTPUT SUBSECTION

It should also be noted, internally the Video Input/Output subsection inputs video data in 16-bit format (packed two samples/word) where it may be placed on the internal ID or IE buss's. Similarly the Output section may output data from the internal ID and IE buss's. Since these buss's are also connected via buffers to the Frame Memory, input data may be easily sent to the Frame Memory and output data may be easily extracted from Frame Memory. Moreover, these data paths selection are software controllable thus allowing for maximum overall system flexibility.

The internal clock of the AAS system will be synchronized to the incoming composite video or separate vertical/horizontal sync signals. Frame rate of 60 hertz ±2% will be accepted. Horizontal and vertical sync will be reconstituted after going through each section of the preprocessors so that frame stabilization will be accomplished.

Sync dropout compensation (B3) will include one horizontal line buffer to store the data of the previous line which will be used in place of the present line in the case of one sync dropout. In case two to eight consecutive syncs are missing, flags will be sent to the configuration controller to delete the processing of those lines for the present field. In case of nine or more missing syncs, the present field will not be processed.

Four configuration controller selectable, gated synchronous start oscillators are provided to accommodate the 525-line and 875-line sampling rates plus two optional rates.

c. Configuration Control Processor. The Configuration Control Processor CCP is designed to generate the basic control and configuration information and signals for the following AAS subsystems:

- Digital Video Preprocessor
- Frame Memory
- Graphics Display Memory

For the DVPP, the CCP controls the following subsections:

- Each Video Preprocessor Module
- Input Video Amplifier
- Internal DVPP Buss Assignments

This control flow is outlined in Figure 3-14.

The Configuration Control Processor receives the basic control information from the AAS Control Processor as shown in Figure 3-15. This control takes place via a buffer RAM memory referred to as Configuration Control RAM (CCRAM).

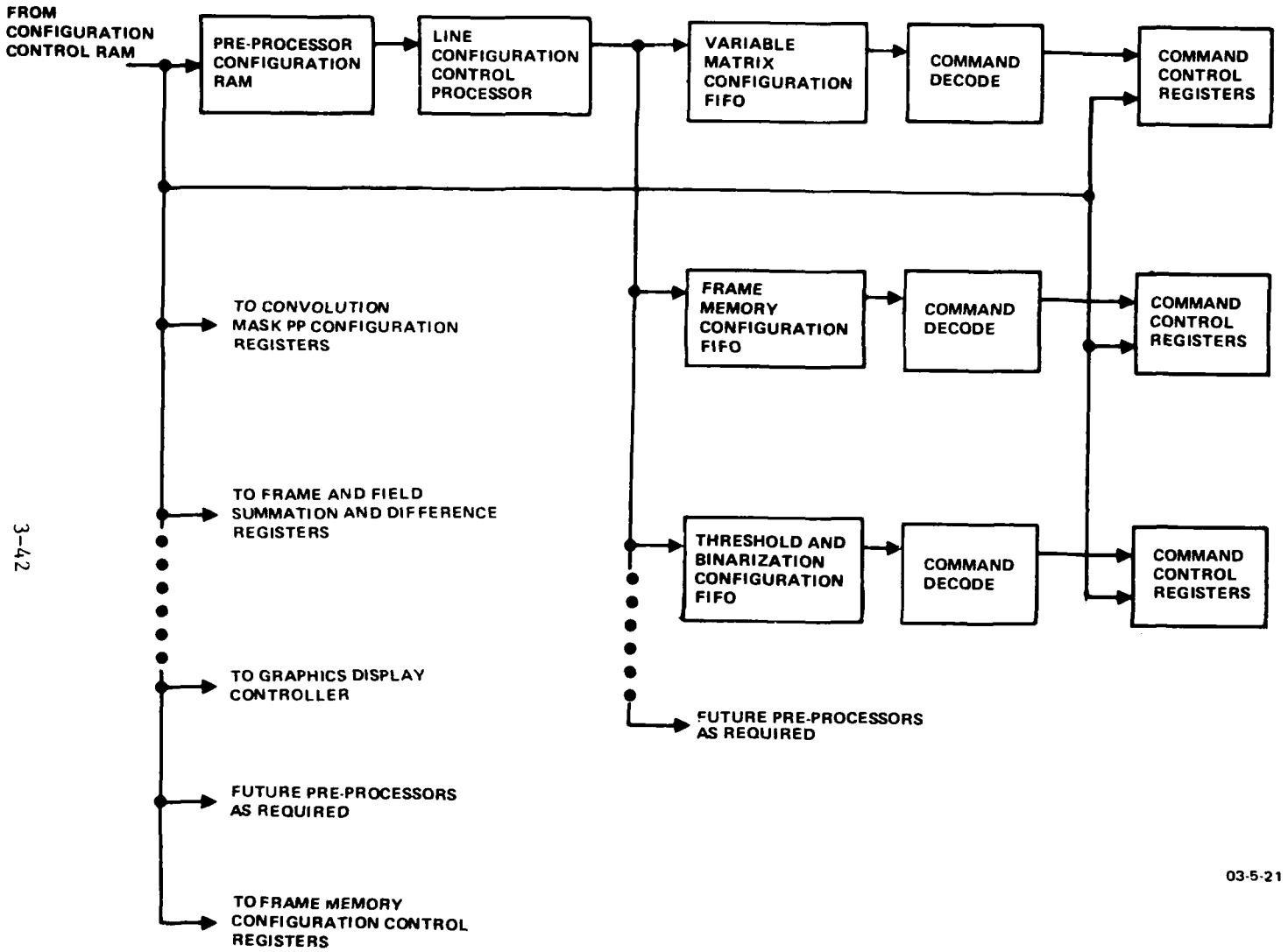


FIGURE 3-14. PREPROCESSOR CONTROL DIAGRAM

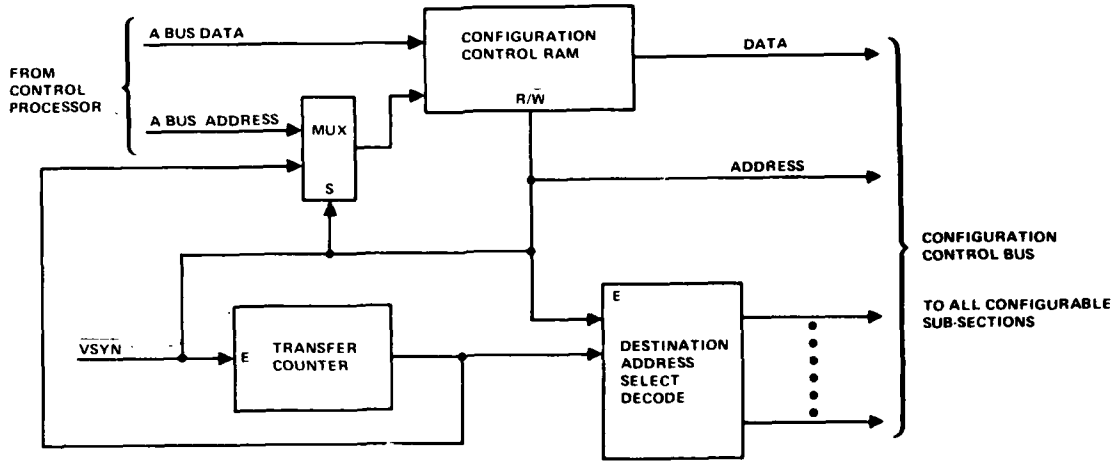


FIGURE 3-15. CONFIGURATION CONTROL INTERCONNECT

The contents of the CCRAM is burst loaded to the configurable subsections, starting with the leading edge of vertical blanking. During this burst load period the CCRAM receives sequential addresses from the transfer counter. During active field time the MVIPS Control Processor is free to randomly write data into the CCRAM via the A Bus. Once the AAS Control Processor has initialized the CCRAM, upon power up, it is only necessary for the AAS Control Processor to update the parameters that require change on a field to field basis, for the parameters that do not require change remain in the CCRAM from field to field.

Most of the front end (preprocessors, etc.,) are also reconfigurable on a line by line basis, to some extent. This line control is provided by a bit slice configuration control processor in much the same manner as the Graphics Control Processor Controls display.

d. Video Preprocessor Subsection (B2). The video preprocessor subsection is the main computing functional element of the DVPP. In general, this subsection is actually a series of processing modules that are designed tc perform functions such as the following in realtime:

- Frame (field) summation/subtraction
- Compass mask

- Laplacian mask
- Roberts mask
- Low pass filter (two dimensional)
- Sobel operator
- Thresholding
- Neighborhood processing (thinning, etc.)

Moreover, the DVPP is designed so that each processor may be incorporated into the design in a module fashion, i.e., all necessary control signals are available for additional preprocessors and other subsections of the DVPP do not have to be modified.

The standard input and output signal format for each preprocessor is diagrammed in Figure 3-16. Also, it should be noted that the selection of the optional input sources and optional destination of the processor output is under the control of the Configuration Control Processor. Included among the possible configurations is the ability to feedback video data that has already been processed by a preprocessor. This feature alone with the capability of modifying the configuration of a preprocessor during vertical retrace time makes the DVPP a very powerful tool for implementing large two-dimensional or recursive filters in almost realtime for the AAS baseline design.

The baseline design for the Video Preprocessor subsection for the Digital Video Preprocessor will contain the following features:

- Preprocessor Video Configuration Control
- Threshold and Binarization Preprocessor
- Variable Matrix Preprocessor
- Video Mask Preprocessor

The features are described in detail as follows.

1. Preprocessor Video Configuration Control. The preprocessors are capable of 25 MHz throughput and are dynamically reconfigurable at field rate. Each can accept its incoming data stream from one of several sources of data, i.e., Input Video Data, another preprocessor or a previously stored frame or field of data. Each of the preprocessors is capable of individually selecting a starting and a stopping point between which it will accept data to be preprocessed, i.e., a row and column number is loaded into the preprocessor from the configuration control RAM, for each the starting point and the stopping point.

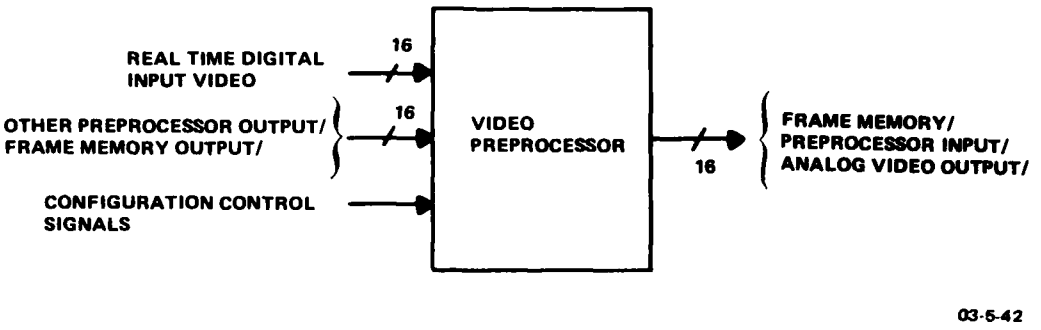


FIGURE 3-16. STANDARD PREPROCESSOR INTERFACE

The preprocessors reconstitute the timing signals input to it, so as to compensate for internal processing time. These reconstituted timing signals, with output data, may then be routed to a second preprocessor which will see data and timing as if it were coming from digitized video input. Utilization of this method of timing and control allows the ability to add additional preprocessors as they become identified. All reconstituted signals are digitally timed and synchronized, i.e., no one shots are utilized. Also, this method provides the ability to input data of different rates without the need to "tweek" the timing and control logic.

All preprocessors may be reconfigured at a field by field rate. Configuration control is received directly from the Configuration Control RAM, in burst mode, each vertical blanking at field. However, certain preprocessors are also reconfigurable at a line by line rate. The line by line configuration control information is passed during vertical blanking, in burst mode, to the preprocessor configuration RAM. The preprocessor configuration RAM is then available during field time to the configuration control processor. This processor, in turn, makes the necessary calculations for control of each of the preprocessors on a line by line basis. Refer to Figure 3-13 for a block diagram.

Default configuration control is provided for all preprocessors. This allows the preprocessor to automatically assume a selected mode of operation for the start of each line as a result of horizontal sync, i.e., most all discrete commands emanating from the command decoder may be selected to be generated from horizontal sync.

2. Threshold and Binarization Preprocessor (TBPP) (B2). The purpose of the TBPP is to implement the function described in Figure 3-17. Where the upper limit (UL) and lower limit (LL) are set by the Configuration Controller, i.e.,

$$\text{pixel value output} = \begin{cases} 0 & \text{pixel in } >UL, \text{ pixel in } <LL \\ 1 \text{ or input data, otherwise} \end{cases}$$

The TBPP also counts the number of pixels over the upper limit, the number under the lower limit and records the x-y coordinates and values of pixels within limits. This information is then sent to frame memory as determined by the mode configuration. (At line, field or frame rate.)

The TBPP function, though very simple, is extremely valuable in the implementation of realtime acquisition algorithms where every field of the data must be examined. By adaptively selecting UL and LL, the input bandwidth of the data may be reduced by orders of magnitude using thresholding techniques.

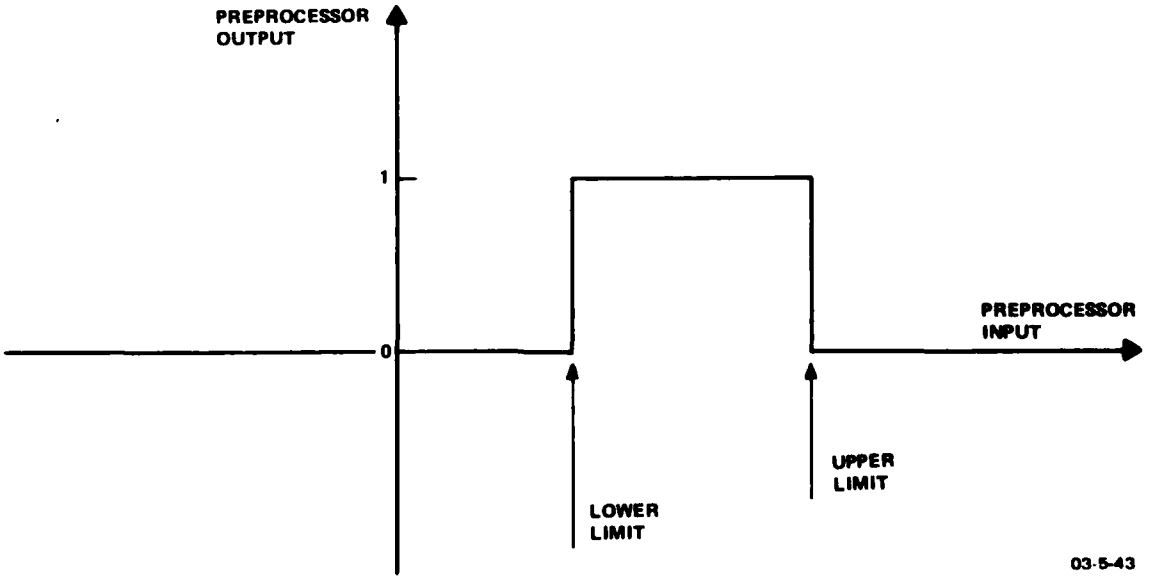


FIGURE 3-17. TBPP PREPROCESSOR FUNCTION

3. Variable Matrix Preprocessor (VMPP) (B2). The VMPP is designed to implement a bank of 1024 low pass filters where each filter output is defined by

$$g_{I,J} = \sum_{I,J} f_{ij}$$

where

$g_{I,J}$ = filter output for grid location I,J

I,J = location of subsection of field/frame

$f_{i,j}$ = pixel within area defined by grid

The grid location and filter size are determined by inputs from the Configuration Controller as shown in Figure 3-18.

The position of the grid relative to the area of interest, defined by First Line, Last Line, First Pixel, Last Pixel, is configured by Pixel Delay and Line Delay configuration. The grid size outlining each matrix is configured by X Matrix Size and Y Matrix Size on a line by line, field or frame basis. This allows variations in grid dimensions within one frame. The spacing of the grid lines horizontal and vertical may be set at 1, 2, 4, 8, 16, 32, or 64 pixels or lines. Thus the dimensions of each matrix may be any combination from 1 x 1 to 64 x 64 pixels.

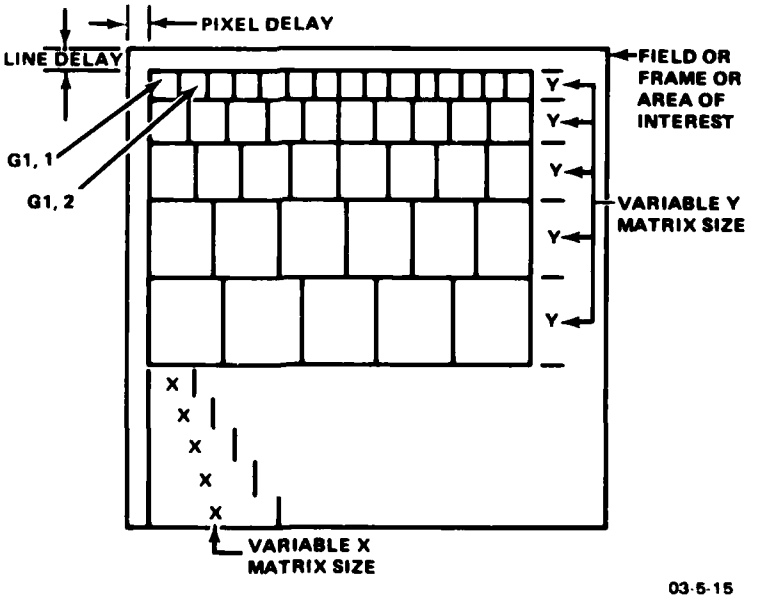


FIGURE 3-18. GRID LOCATION AND FILTER SIZE

The VMPP module contains 1024 holding registers, each 20 bits wide. One holding register is assigned to each matrix within the grid pattern. As each pixel comes in, its coordinates identify the holding register and the pixel value is added to the contents of its associated register. At the end of the field, frame or matrix line the contents of each register is scaled from 0 to 12 places as determined by configuration and an 8 bit value for each matrix is sent to memory.

4. Sum and Difference Preprocessor (SDPP). The function of the SDPP is to simply compute the sum or difference for/between the output of the video A-to-D converter and the feedback input to the processor from frame memory.

The operation mode is either the sum of the two pixel values divided by two or the difference between the two values times two. The result of this computation is output for distribution according to the Configuration Controller

5. Video Mask Preprocessor (VMP). The purpose of the Video Mask Preprocessor is to implement the 3 x 3 convolution operation in realtime for cases where coefficient in the convolution kernel are powers of 2. This may at first appear rather restrictive, however a large class of common operations fall into the resulting class, e.g. compass gradient masks, Laplacian, Sobel, and Roberts. Moreover, this restriction eliminates the requirement for high speed multipliers for the realtime implementation. The resulting programmable logic arithmetic (PLA) lends itself readily to a VLSI implementation with a standard cell being repeated many times within the design.

The Video Mask is implemented as follows: A 3 x 3 pixel array (Figures 3-19 and 3-20) is maintained in real time and under configuration control perform arithmetic operations with a running overlap on the array as it advances through the field.

Every pixel time the 3 x 3 array is as follows:

The nine pixel values are entered into a pipelined arithmetic pyramid. This arithmetic pyramid is a nine step process so at a given point in time nine sequential pixel arrays are being processed at pixel rate in real time. An example of this arithmetic pyramid is shown in Figure 3-21.

The arithmetic pyramid contains 48 arithmetic units (some can multiply or divide by 2, 4, 8, 16, etc., some can add or subtract) that are individually programmed by the Configuration Control Processor at vertical sync time. This makes the Video Mask a very useful and versatile module. Examples of typical masks are shown in Figure 3-22.

3.2.2.4 Frame Memory

a. Frame Memory Module (B4). The frame memory is organized as four frames accessible to the multiport controller. Each page of memory contains storage for 256K bytes, e.g. a full frame of 512 x 512 video data. The configuration of the four frames is completely controlled by external sources to the memory.

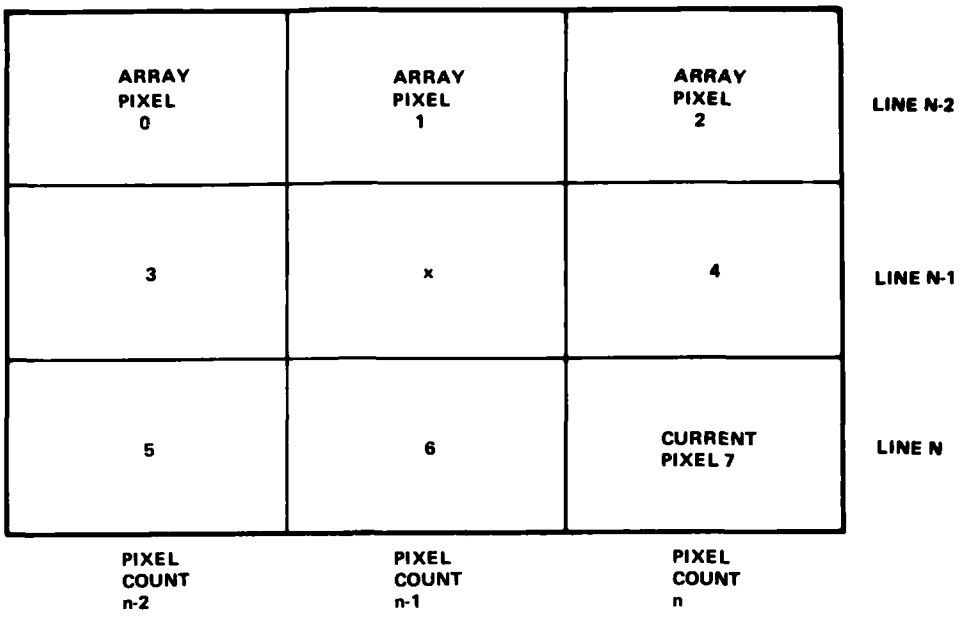


FIGURE 3-19. PIXEL ARRAY

During the vertical blanking time, the configuration control processor is designed to handle the memory assignments via the multiport controller. Any one of the four frames can be assigned to any one bus, any of the remaining three frames can be assigned to any other bus. This configuration can occur during each vertical blanking, so the entire assignment of frames to buses can be revised after each field of data. This architectural approach gives the memory great flexibility. This scheme allows frames of data to be input from one source, accessed by other sources for processing and output to another unit without complicated addressing changes. All that is needed to control memory assignment is a few bits on the configuration control bus.

There is also considerable flexibility built into the addressing of the memory. During the vertical blanking time the configuration control processor will send data to control the addressing. Four parameters are sent for addressing--starting x, y and final xy address. The starting address is the memory location to be first accessed after the next start-of-frame. This scheme of addressing greatly simplifies the handling of data. Any portion of the data can be accessed without requiring separate addressing for each byte. The data is transferred immediately and the arithmetic units do not require separate addressing logic. This addressing scheme allows "windows" of data to be read from a frame without disturbing the original frame data. It also allows any interlace pattern to be easily accomplished.

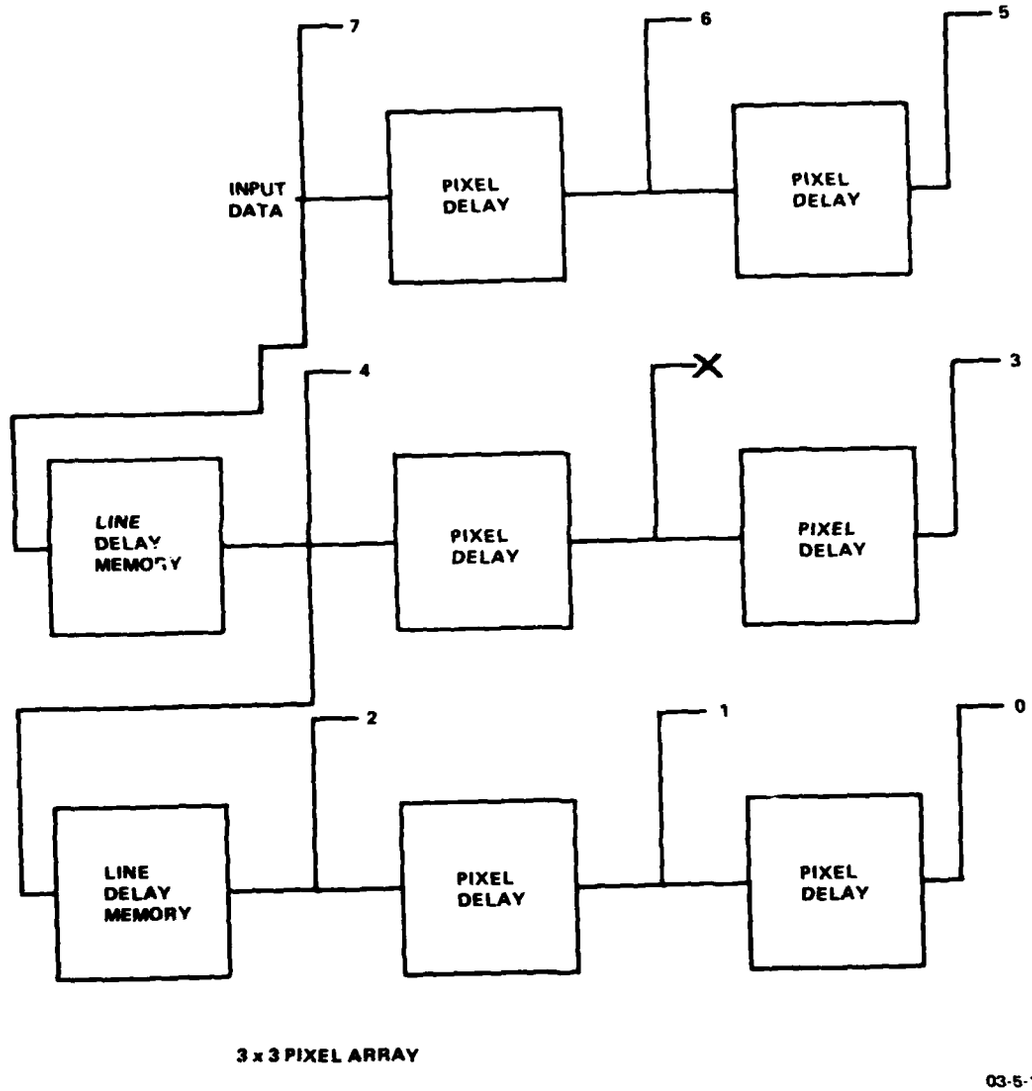


FIGURE 3-20. VIDEO MASK 3X3 PIXEL ARRAY

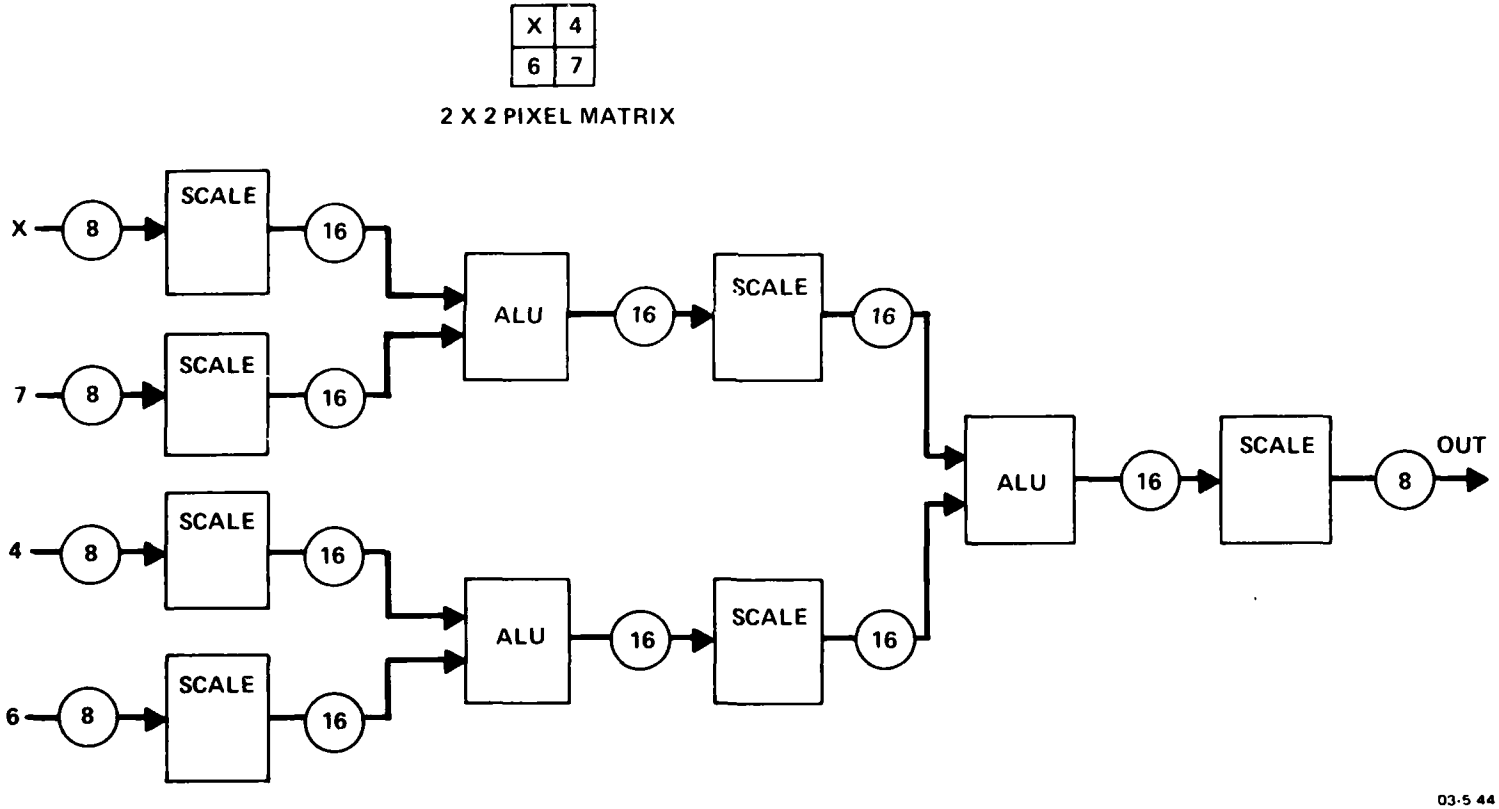


FIGURE 3-21. ARITHMETIC PYRAMID

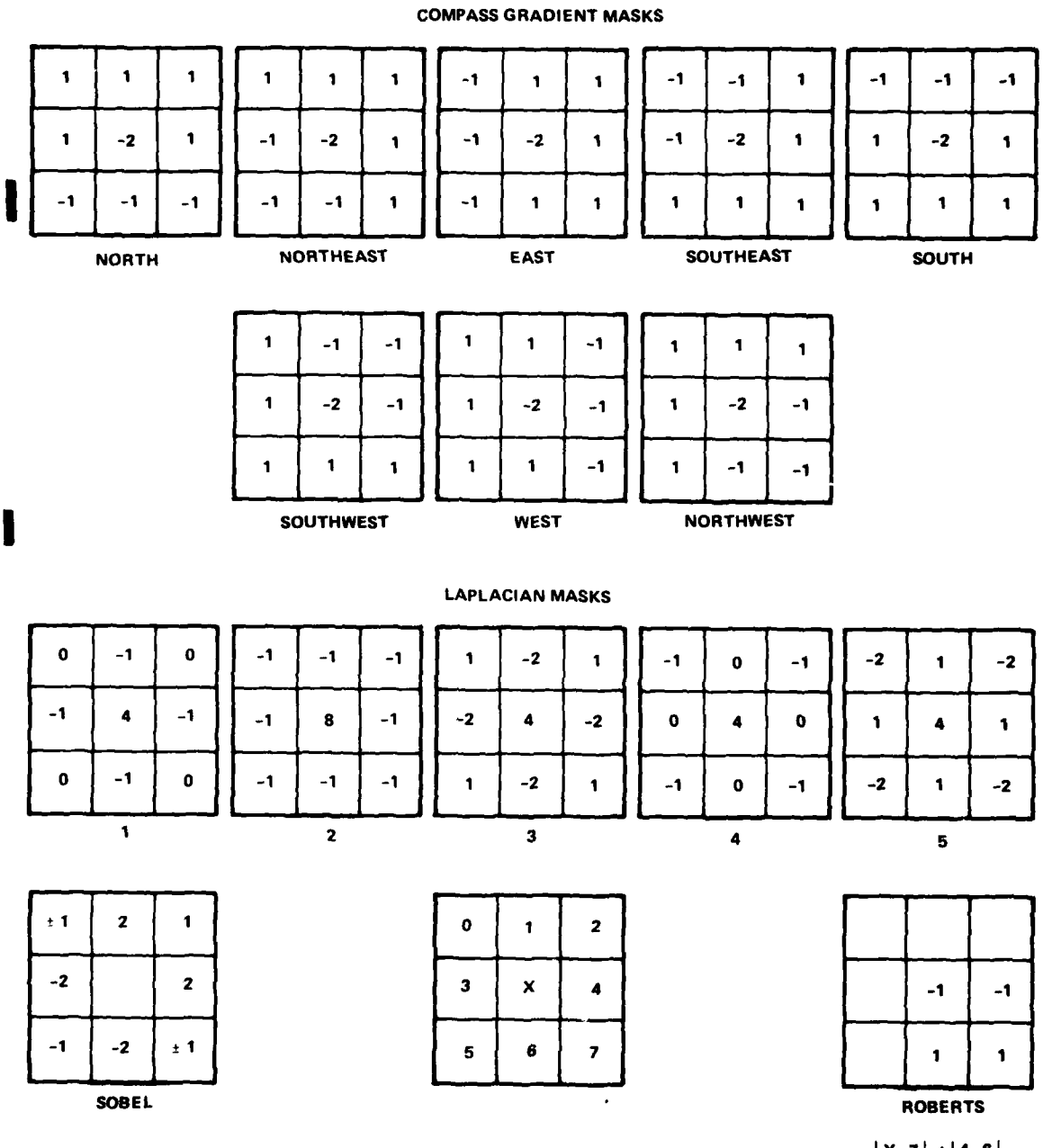


FIGURE 3-22. EXAMPLES OF VIDEO MASKS

The memory has been designed so that it can be expanded in capacity. The board is designed for 1 megabyte of memory. The basic operation of the memory will be unaffected by the expansion, there will still be four frames assignable to four buses. The memory addressing will require additional bits to accommodate the expansion.

b. Multiport memory interface control. The multiport memory interface controller provides the steering logic and control of the memory modules. This control is performed under the direction of Configuration Control, during vertical blank time.

The steering logic provides the ability to connect any one of the four frame memories to either of the two video internal data busses (Bus ID or IE) or either of the two processor busses (Bus B or C).

The two internal busses (Bus ID & IE) do not have the ability to address memory, and therefore the Configuration Control, via the multiport memory interface control, controls the locations at which data is stored. This method of memory addressing allows for the most efficient means of utilizing memory - both from the standpoint of access time and memory size.

3.2.2.5 High Speed Digital Microprocessor. The High Speed Digital Microprocessor is designed to meet the computational and data handling requirements for acquisition algorithms that in general require a large number of operations per pixel and which may not be easily implemented with dedicated digital pipeline hardware such as in the Digital Video Preprocessor. The typical algorithms that the HSDM is designed to implement are as follows.

(1) Small to large two-dimensional recursive and nonrecursive filter

(2) Complex multiple task algorithm that may require concurrent processing e.g. intelligent target tracking algorithms, auto-screening/cueing algorithms, and intelligent image bandwidth compression algorithms.

(3) Perform high speed (nanosecond) control and data transfer functions

In accordance with the objectives of FACC's internally funded program, the architecture for the HSDM is designed to implement these algorithms, in particular the auto-screening/cueing algorithm, in realtime as might be required for an operation system. This architecture as shown in Figure 3-23 is in fact the architecture distributed processor that include both a SIMD and MIMD structure.

The baseline configuration of the HSDM consists of a Control Processor (CP), a Microprogram Sequencer (MS), an Address Generator (AG), and a Pipelined Arithmetic Processor (PAP). A detailed description of each of the elements is given in the following sections.

a. Control Processor (B.4) The Control Processor (CP) is a sixteen bit, general purpose, microprogrammed machine using state-of-the-art LSI components. A vectored interrupt structure is provided to allow efficient communication with external devices. Two sixteen bit buses (the I/O Address bus (CPA) and the I/O Data bus (CPD)) are used to interface with external devices. The CP includes a scratch pad memory for storage of parameters and intermediate results, and for I/O buffers. A hardware multiplier is provided for increased arithmetic capability. A programmable timer is used to create interrupts at programmable intervals. The block diagram for the CP is shown in Figure 3-24.

The CP microinstruction word length is 64 bits, microinstructions are executed at a 7 MHz rate. Register to register type macro instructions are executed in one micro-cycle while a conditional jump instruction typically requires four micro-cycles.

1. Functional Description. The CP consists of two main sections, the program control section, and the ALU section. The program control section consists of those elements which affect the flow of control at the macro instruction level or at the micro instruction level. Those elements, as shown in Figure 3-24 are

- Program sequencer
- Program memory
- Instruction register
- Microprogram sequencer

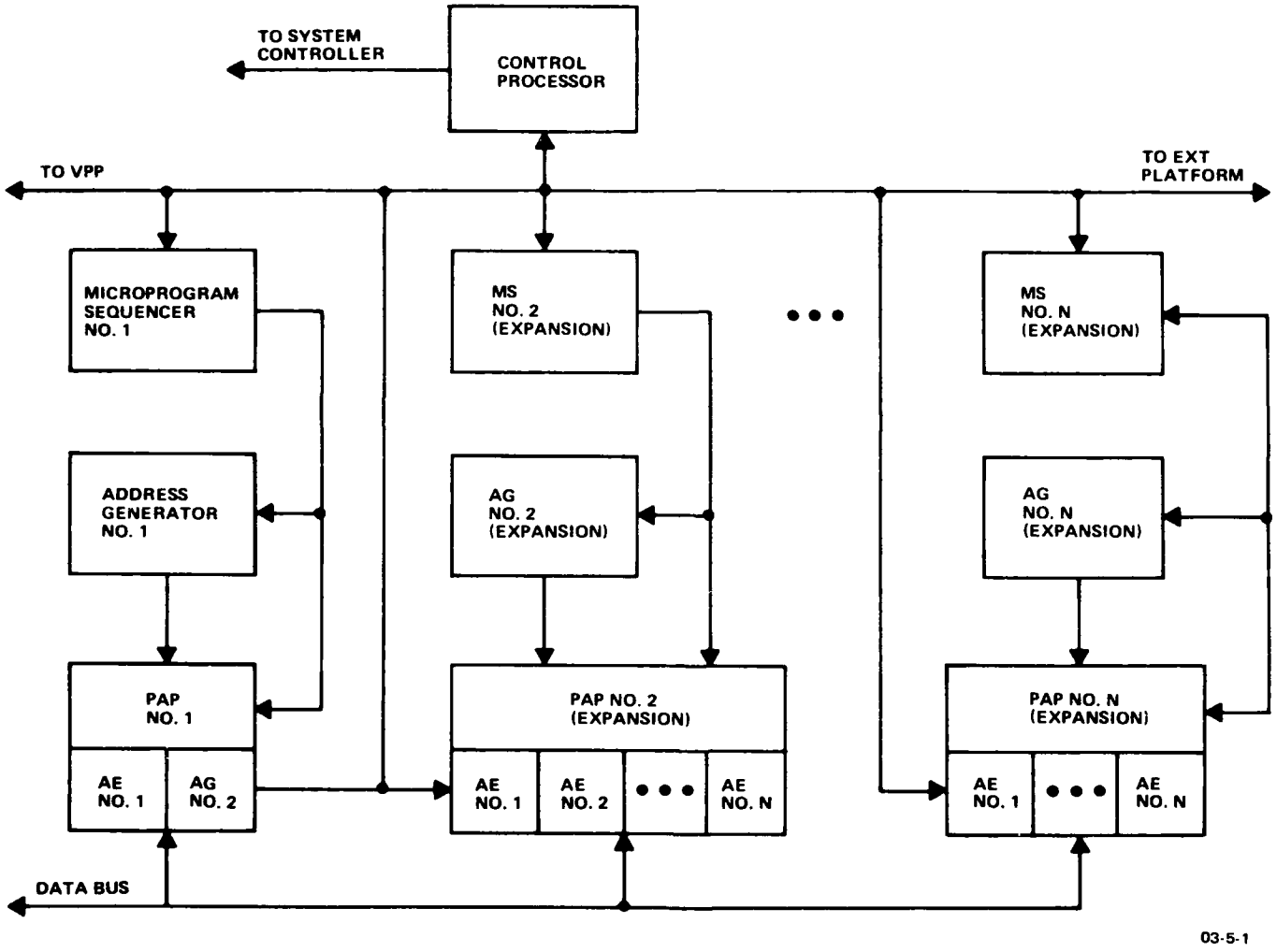


FIGURE 3-23. HIGH SPEED DIGITAL MICROPROCESSOR BLOCK DIAGRAM

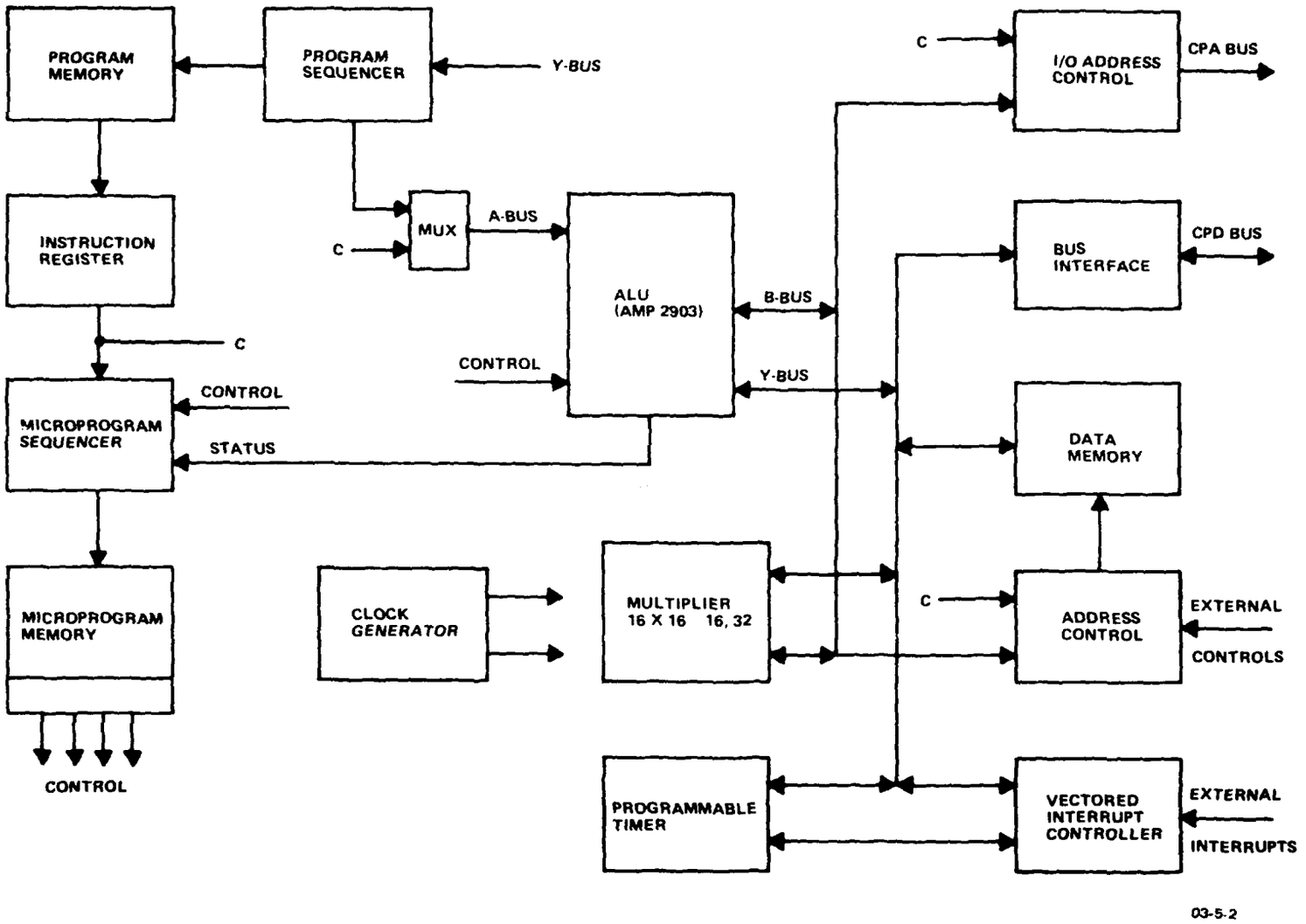


FIGURE 3-24. AAS CONTROL PROCESSOR BLOCK DIAGRAM

- Microprogram memory
- Vectored interrupt controller
- Programmable timer

The ALU section consists of those elements which perform data manipulation and data transfers. These elements are:

ALU

Multiplier

Data memory

Address control

Bus interface

I/O address control

2. Program Sequencer and Program Memory. The Program Memory (PM) contains the CP instructions. In the AAS system the CP has 32K words of memory, but provisions for expansion up to 64K words are provided. Program Sequencer (PS) controls the selection of instructions from the PM. The PS selects instructions sequentially from the PM until an instruction is encountered that selects an instruction to be executed other than the next sequential instruction.

3. Instruction Register. The Instruction Register (IR) receives instruction words from the program memory. These instruction words are normally used by the microprogram sequencer to select the appropriate microinstruction sequence. For double word instructions, however, the second word is used as an input to the ALU rather than the microprogram sequencer. This allows jump addresses and immediate constants to be contained in the program memory.

4. CP Microprogram Sequencer and Microprogram Memory. The Microprogram Sequencer (MPS) controls the selection of micro-instructions from the Microprogram Memory (MPM). Macro instructions in the IR are used by the MPS to select the micro-instruction sequence required to accomplish the macro-instruction. The micro-instruction sequence may contain a single micro-instruction (for example, register to register operations). Conditional micro-instructions are implemented which allow decisions to be made dependent on the status from the ALU.

5. Vectored Interrupt Controller. The Vectored Interrupt Controller (VIC) allows the program flow to be modified by external events not related to program executions (for example the programmable timer interrupt). The CP design allows the VIC to be controlled either at micro instruction level, the macro instruction level, or a combination of the two.

6. Programmable Timer. The Programmable Timer (PT) allows repetitive interrupts to be generated at a programmable rate.

7. ALU. The ALU is a sixteen bit unit consisting of 2903 bit slices and carry lookahead logic. The ALU provides the capability to perform arithmetic and logical functions. Also included in the ALU is a dual port register file for data storage. The ALU provides the necessary logic to perform the divide function. The ALU has two bidirectional buses and one unidirectional bus for receiving and transmitting data.

8. Multiplier. The multiplier is a TRW 16 by 16 bit multiplier (MPY16HJ). The multiplier provides a 32 bit product in two micro-cycles.

10. Bus Interface and I/O Address Control. The Bus Interface and I/O Address Control provide communication with external devices over the CPA bus and the CPD bus. The CPD bus is bidirectional as is the Bus Interface. The internal Y BUS may be transmitted to the CPD bus or the CPD bus may be transmitted to the Y BUS. The CPA bus is unidirectional and is driven by the I/O Address control. There are two sources for the CPA bus: (1) the ALU B-BUS; or (2) the immediate operand of a two word instruction (from the IR).

b. Microprogram Sequencer (B.6). The Microprogram Sequencer (MS) is the Microprogram address/instruction generator. It generates 16 bit addresses to and receives 96 bit words from the Microprogram memory. The MS and the Address Generator are initialized for each macro instruction by the Control Processor via the Macro Queue. Multiple macro instructions may be stored in the Macro Queue to be performed sequentially without disrupting the Control Processor. Refer to Figure 3-25 for a block diagram of the MS.

1. Physical Description. The MS contains 130 integrated circuits. The MS logic is implemented in TTL and ECL circuits and uses 35 watts of power.

2. Detail Description. The basic blocks of the MS are:

(1) Macro Queue

(2) Loop Counters

(3) Address Generators and Selection

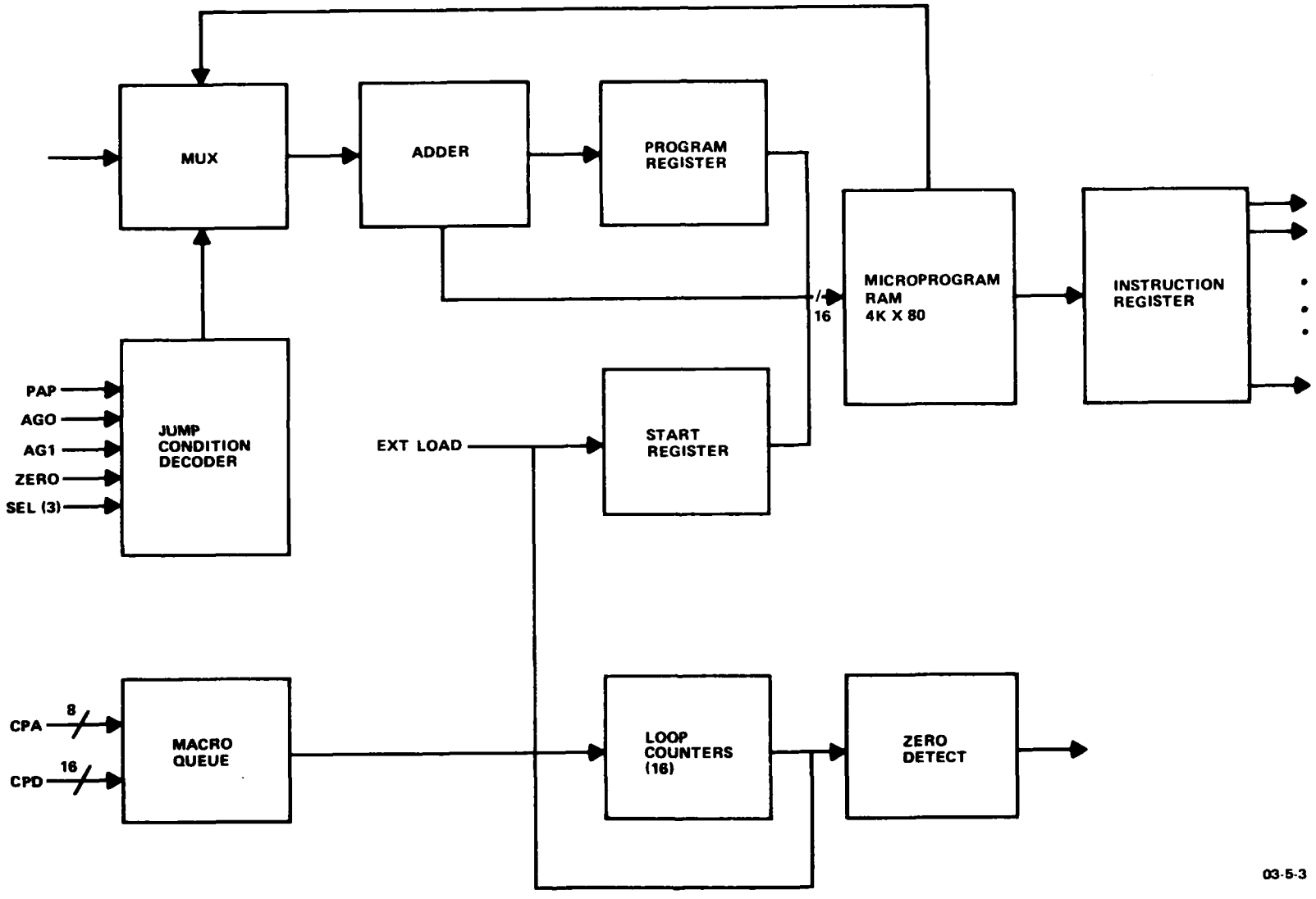


FIGURE 3-25. MICROPROGRAM SEQUENCER

(4) Microprogram Word

(5) Control

The Macro Queue is a 24 x 128 FIFO that is loaded by the Control Processor with 16 bits from the CPD bus and 8 bits from the CPA bus. The bits are defined as follows:

CPD 0 → 15 - Data

CPA 0 → 3 - Address - either Address Generator or Microsequencer counter

CPA Bits	7	6	5	4	
	0	0	1	0	Address Generator 2901 #0
	0	0	1	1	Address Generator 2901 #1
	1	0	0	0	Microsequencer Counter
	1	1	0	0	Microsequencer Program Register
	0	0	0	0	Run

An "idle" signal from the Microsequencer allows the Macro Queue to transfer data to the designated device. Run is the last instruction in each Macroinstruction. Several sets of instructions may be queued in the FIFO.

The Maco Queue may be cleared and reloaded at any time by the Control Processor.

The 16 Loop counters and 16 holding registers are loaded with 16 bit words from the Macro Queue. As each instruction is executed the counter addressed by 4 bits of the Microprogram word is decremented. When the count in any counter reaches zero it is reloaded with its initial count from the associated holding register.

The first address of the Macroinstruction is loaded from the Macro Queue into the program register. When a "Run" signal is received from the Macro Queue the multiplexer looks to the program register for the first address. For subsequent addresses the Multiplexer locks to either the adder or the counter. The adder adds the 12 bit 2's complement number from the microprogram word to the 12 bit address in the program register. The counter provides an address of the present address plus 1.

Except for the first instruction the multiplexer will look to the adder for the address unless a condition identified by four bits in the microprogram word is satisfied. In that case the multiplexer will look to the counter for the next sequential address.

The 20 LSB in the Microprogram word are used by the MS. They are defined as follows:

0 → 11 Jump Count

19	18	17	16		
0	0	0	0	=	Unconditional jump
0	0	0	1	=	Increment if addressed counter = 0
0	0	1	0	=	Increment if AD 0 flag true
0	0	1	1	=	Increment if AD 1 flag true
0	1	0	0	=	Increment if PAP flag true
0	1	0	1	=	Increment if AD 1 FIFO is full
0	1	1	0	=	Increment if Data Ready signal
0	1	1	1	=	End of Macro-Reload
1	0	0	0	=	Subroutine jump*
1	0	0	1	=	Return from Subroutine*
1	0	1	0	=	Spare* 1101 = Spare*
1	0	1	1	=	Spare* 1110 = Spare*
1	1	0	0	=	Spare* 1111 = Spare*

15 - 12 Counter Address

The control section consists of the miscellaneous circuits for instruction decoding, time and control signal generation and issuing interrupt and flags to the rest of the system.

c. Address Generator. The Address Generator provides the two sixteen bit addresses used by the pipelined arithmetic processors and 20 bit address for the frame memory. The addresses are generated from the two sixteen bit high speed processors. The two processors are capable of operating independently of one another. Programming of the two processors is provided by a forty bit micro-instruction word. An additional address control feature is provided by a skip command embedded in the micro-instruction word. The skip command selects one of the two processor outputs for testing with a fixed constant. The address is tested for greater than, less than, or equal to functions.

1. Physical Characteristics. The address generator contains approximately 90 integrated circuits, and uses 30 watts of power. It executes instructions at x 10 Mhz clock rate.

2. Detailed Description. The address generator can be divided into three distinct functions.

(1) Instruction Registers and Decoding

(2) Arithmetic Processor

(3) Skip Command Logic

Each of these functions are discussed in detail in the sections following. A simplified block diagram of the address generator is shown in Figure 3-26.

The addresses from the Address Generator are generated from a forty-six bit micro-instruction word and three control signals. The micro-instruction are latched to assure time for decoding of commands and directing of data. Pipeline selection is determined by the three control signals; RUN/HALT, IDLE, and Macro Enable. The three controls enable four distinct modes of operation; RUN, HALT, IDLE, and INITIALIZATION. The processors are on the RUN mode if none of the three control signals are activated. The HALT mode is a front panel activated signal which allows the user READ only access to the sixteen internal registers of one of the address processors. In the IDLE mode both address processors are in a "NOOP" state and an address of zero is present. The INITIALIZATION mode is a selectable preset given at the start of a macro instruction string.

Each address generating processor is a high speed sixteen bit microprogrammed arithmetic processor. The two processors use a 2901B bit-slice microprocessor as its basic building block. Data to each processor is selectable from one of two data sources, external input or data from micro-instruction word. The processors can execute 16-arithmetic operations as well as perform data transfer.

The skip control logic allows for the output of either address processor to be tested for greater than, less than, or equal to conditions, when compared to a test word. When the condition is satisfied, the skip flag is set preventing the subsequent micro-instruction word from changing the contents of the processor's internal registers. The skip flag remains set until the selected condition is not met, after which the processor continues normal operation. Six bits within the micro-instruction are used for the controlling of the skip flag and testing word.

The address generator is programmed via a 46 bit micro-instruction word. The micro-instruction word is divided into several fields. For details of the field refer to Table 3-5. Each processor is provided its individual sixteen bit micro-code word. In addition the processor is also capable of passing a sixteen bit data word from memory to the address latches. The A and B address lines of CPU1 are used as the most significant eight bit byte of the data word. The low order byte is the A and B address lines for CPU1. The use of the A and B address line for both processors in affect causes the program to pass a data word through only one processor at a time. The data select bits are used to select either the sixteen bit data word just described or an external data word prescribed by the user. There are six bits dedicated to the skip logic. Each central processor has two bits to determine the manner in which the skip logic is used. The table below gives a functional description of the micro-instruction word.

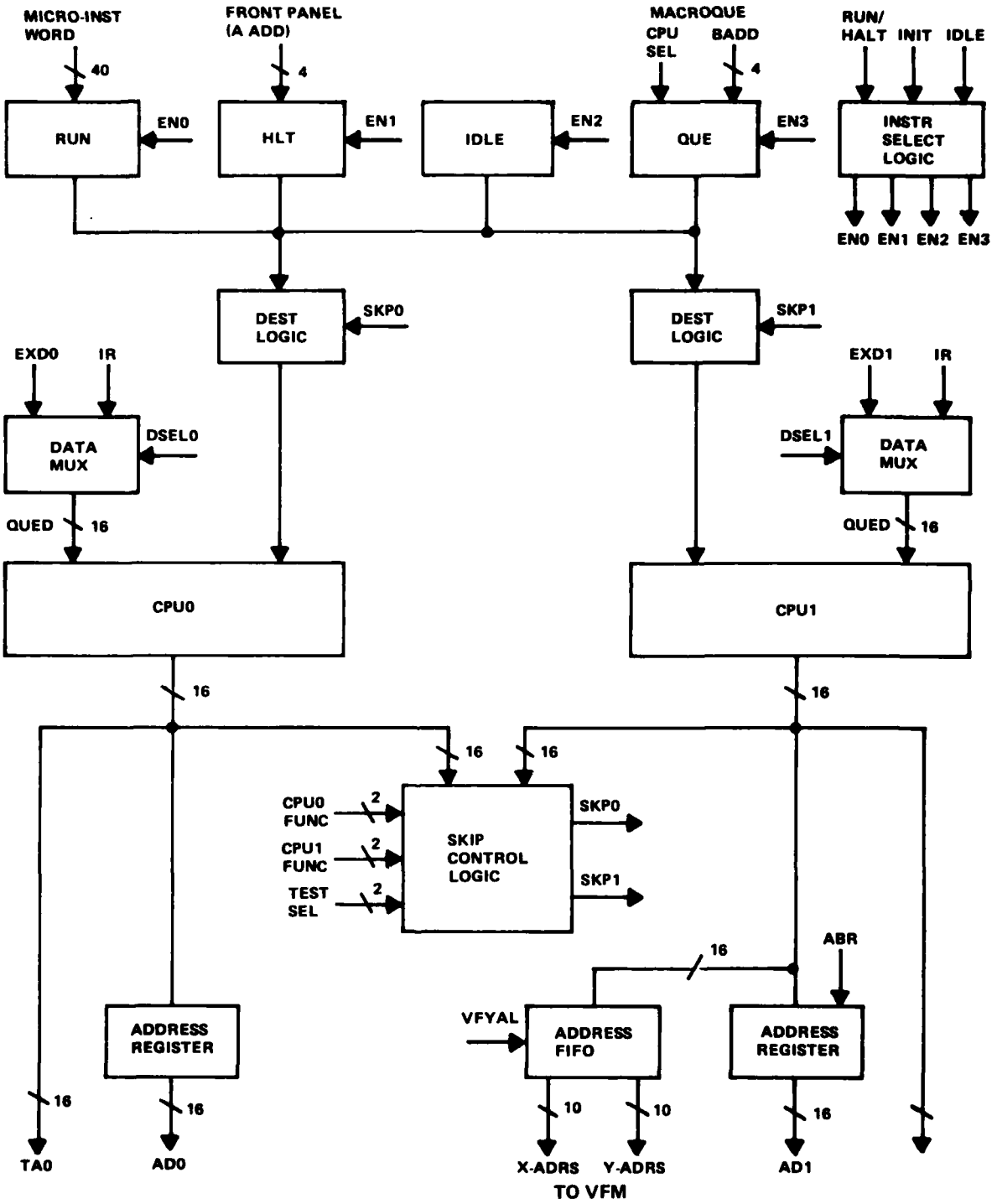


FIGURE 3-26. ADDRESS GENERATOR BLOCK DIAGRAM

TABLE 3-5 ADDRESS GENERATOR

BIT POSITION	
50 - 51	CPU1 - condition of skip logic
52 - 53	CPU0 - condition of skip logic
54 - 55	Test being performed for skipping
56	CPU1 - data select
58	CPU0 - data select
60 - 63	CPU1 - B register address
64 - 67	CPU1 - A register address
68 - 71	CPU0 - B register address
72 - 75	CPU0 - A register address
76	Write address FIFO
77 - 85	CPU1 - Instruction field
86 - 95	CPU0 - Instruction field

d. Pipelined Arithmetic Processor (B.4). The Pipelined Arithmetic Processor (PAP is a high speed, programmable arithmetic logic module). The PAP executes the required arithmetic instructions to solve the image processing algorithms. The PAP is broken down to Arithmetic Elements (AE) with independent hardware and sharing only control functions. For a block diagram representation of the PAP refer to Figure 3-27.

1. Physical Characteristics. The PAP contains 50 IC's for the control logic, 100 IC's for each AE, and 20 for test for a total of 170 IC's. The power used is 35 watts.

2. Detailed Description. The major elements of the PAP are its AE's and the necessary control functions to drive and access them. The major elements of the AE's are: Cache input memories, scratch pad, ALU, Multiplier, and Skip Logic. A description of each follows.

Image processing inputs to the AE are provided by the cache memories. The cache memories are divided into two fully independent memories: cache A and cache B. Each memory has two sides, between which the incoming data is ping-ponged. Ping-ponging increases the continuity and hence the speed of data processing. For example, one side can be used by the PAP while the other side is being refreshed with new data. The manner in which the ping-ponging is done can also be fully programmed at the onset of each individual macro. The present caches have the capability of storing up to 4K x 8 bits of data.

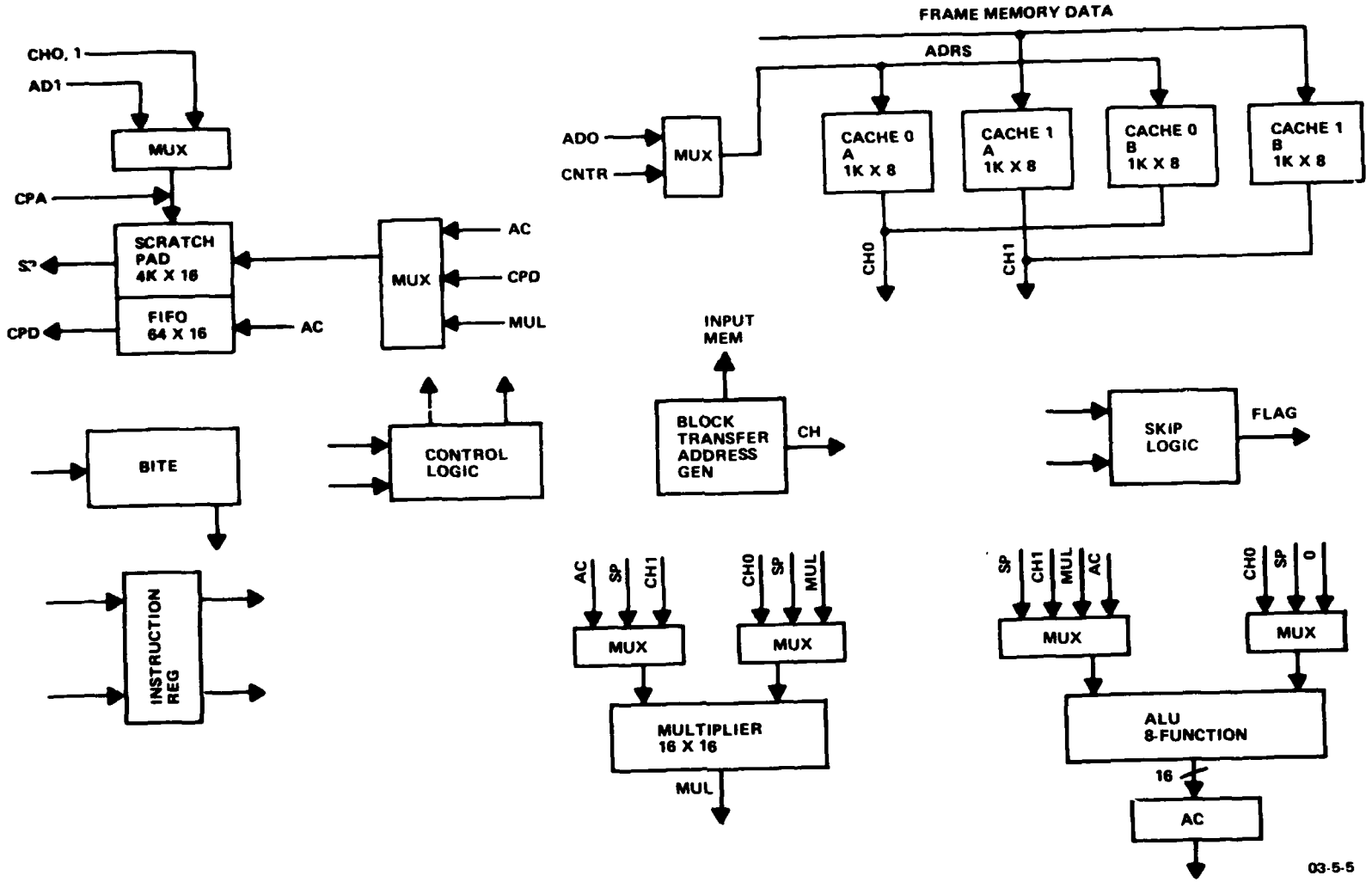


FIGURE 3-27. PAP BLOCK DIAGRAM

A scratch pad memory is an additional requirement for the AE. A scratch pad is a memory used to store the intermediate results of the arithmetic logic units. The scratch pad has the option to accept data from its ALU accumulator, multiplier or the control processor. The addresses that the scratch pad will use are also completely selectable in that the scratch pad can use its independent address generator address or its independent cache output data address. These cache output data addresses are used for making quick histograms where the processor needs to know the frequency of occurrence vs. the data value. Additional flexibility is obtained by using a skip flag to enable a write into the scratch pad. This skip flag increases the capability of the PAP by enabling it to easily do comparisons and true/false logic. The scratch pad can presently store 4K x 16 bits each but they may be expanded to 64K words. Some of this storage may also be used for PROM lookup tables where log, exp, sin, cos, etc... can be stored.

In addition to the scratch pad, there is also a FIFO which provides the asynchronous communication from the AE (PAP) to the control processor. The FIFO stores and transfers data under program control.

The digital processing of the AE is done by the ALU. The ALU performs such functions as: EOR, A or B, A minus B, B minus A, A plus B, hold accumulator, and preset accumulator. Each operand is independently selectable. The logic surrounding the ALU permits double precision (32 bit) arithmetic to be performed under software control.

To further increase the capability of the AE, a TRW multiplier has been included. The multiplier has the capability of multiplying any one of nine permutations of operands. The multiplier also has provisions for outputting a thirty-two bit product, although normally only sixteen bits would be used. The selection is under software control.

Additional program flexibility has also been gained by adding skip logic. The skip logic sets a flag if the accumulator content meets the negative, positive, or zero condition that has been selected. The flag is then used for conditional writes into the scratch pad. The skip flag may also be used by the Address Sequencer to execute conditional skip or jumps through the microcode.

The programming of the PAP is complemented by a twenty-seven bit instruction word. This instruction word is common to the AE's. This feature allows simultaneous processing of independent portions of a given frame.

The micro-instruction word for the PAP is separated into fields, the ALU, Multiplier, Scratch Pad, Skip Logic and FIFO Control Field. Detailed description of the various control fields is given in section 3.2.3.

e. System Interface. The system interface module will control all information being passed between the control panel and the AAS system. The 6802 microprocessor will be the major building block of the system interface module.

The system interface module will consist of all memory and circuitry necessary for executing control panel commands. A 1553 serial interface will be used between the system interface module and the control panel. Command information passed from the control panel will be interpreted and sent to the AAS system along an eight-bit bi-directional data bus and a sixteen-bit address with appropriate bus control signals. In a system type operation, the system interface module will communicate with the system controller, LSI 11/23. In a debugging operation, the system interface module will be capable of monitoring parameters and examining microcode within any of the AAS processors, i.e., video preprocessor, control processor, system controller, and the high speed arithmetic processor. A simplified block diagram of the system interface module is given in Figure 3-28.

3.2.2.6 Graphics Display Controller (B.5). The Graphics Display Controller (GDC) provides the ability to display target gates and cross hairs with identification and text data. Also the GDC controls area blanking, reverse video, reverse graphics, blinking, intensity of graphics output and highlighting of selected video by subduing surrounding video data.

Any type of symbol, character, etc., of any size, in any location within the area of the monitor may be displayed. See Figure 3-29.

The following example illustrates a Target Designator Gate (TDG) and Boresight Crosshair (BSCH) display generation. The TDG is generated by start of a line on Row 1, Column 2 and stopping the line when Column 8 is reached. When rows 2, 3 and 4 are scanned, dots are generated at Columns 2 and 8 thus displaying vertical lines. At Row 5, Column 2, a line is started, but unlike the line "drawn" on Row 1, the line is stopped at Column 4 and a text character is inserted to identify the TDG and again the line is started at Column 5 and then stopped at Column 8. The BSCH may be generated by a similar procedure.

Any number of gates, crosshairs and symbols may be displayed. The only criteria is that any of graphics must be capable of being described (output dot, start line, etc.,) in 64 discrete commands/line. Also, the number of calculations made by the Graphics Display Processor is limited to approximately 500 instructions per line.

The Graphics Display Processor is also capable of displaying overlapping gates as shown in Figure 3-30.

The functional organization of the Graphics Display Process is shown in Figure 3-31 and is described as follows. The Graphics Display is burst loaded from the Configuration Control RAM at the leading edge of the vertical blanking of the display monitor. The data loaded into the RAM contains all of the parameters to describe the graphics display i.e., position in X Y coordinates, size in pixel and line counts, type of symbol or character, intensity, reverse video, reverse graphics, blank video, etc. After completion of the burst load of the Graphics Display RAM the Graphics Display

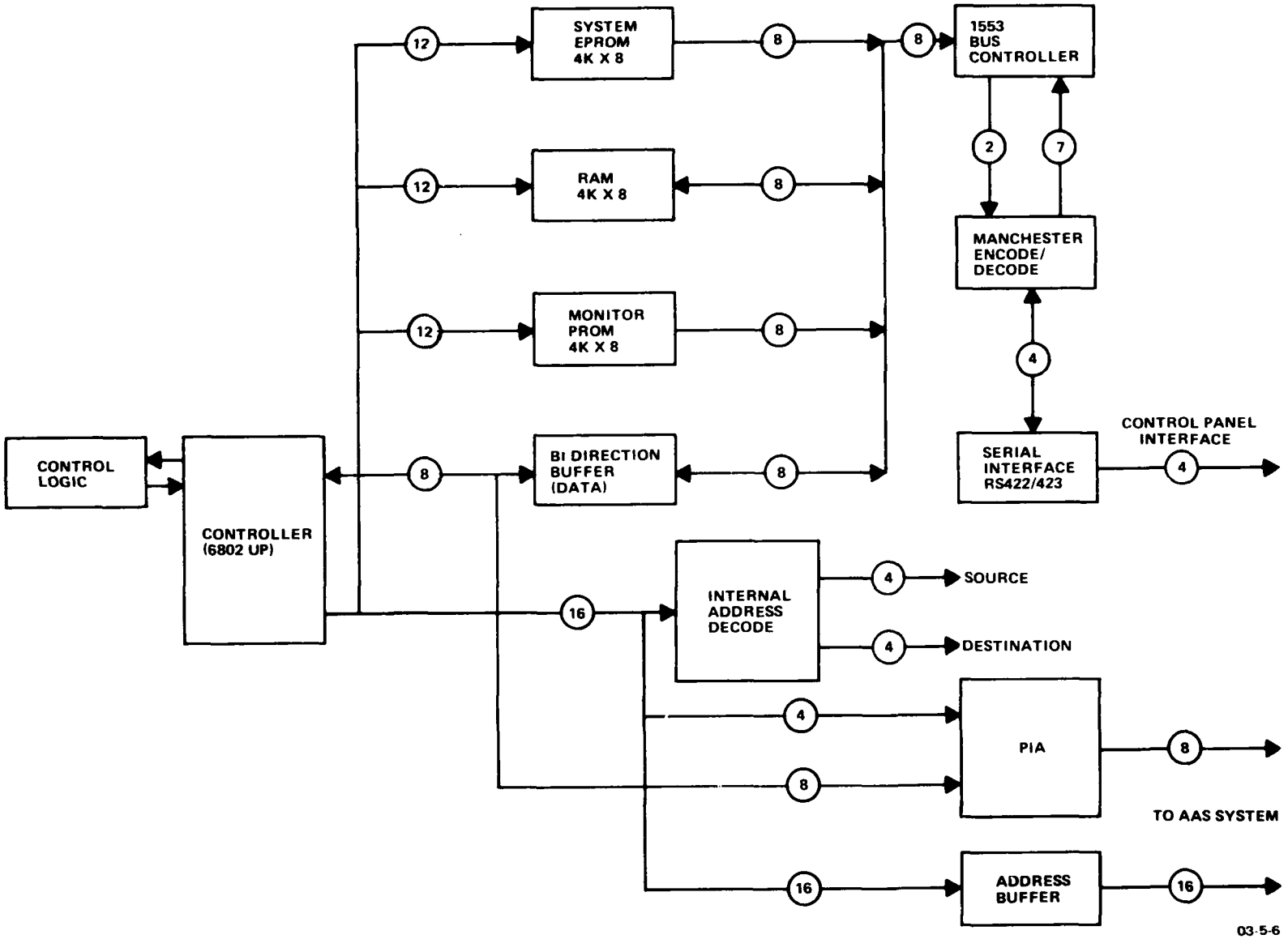


FIGURE 3-28. SYSTEM INTERFACE BLOCK DIAGRAM

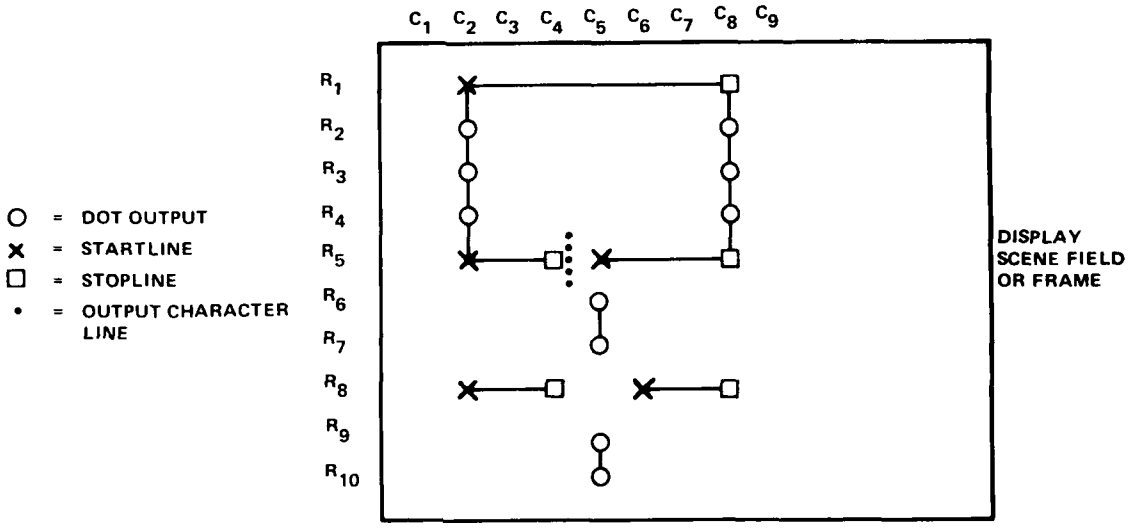


FIGURE 3-29. GRAPHICS DISPLAY GENERATION EXAMPLE

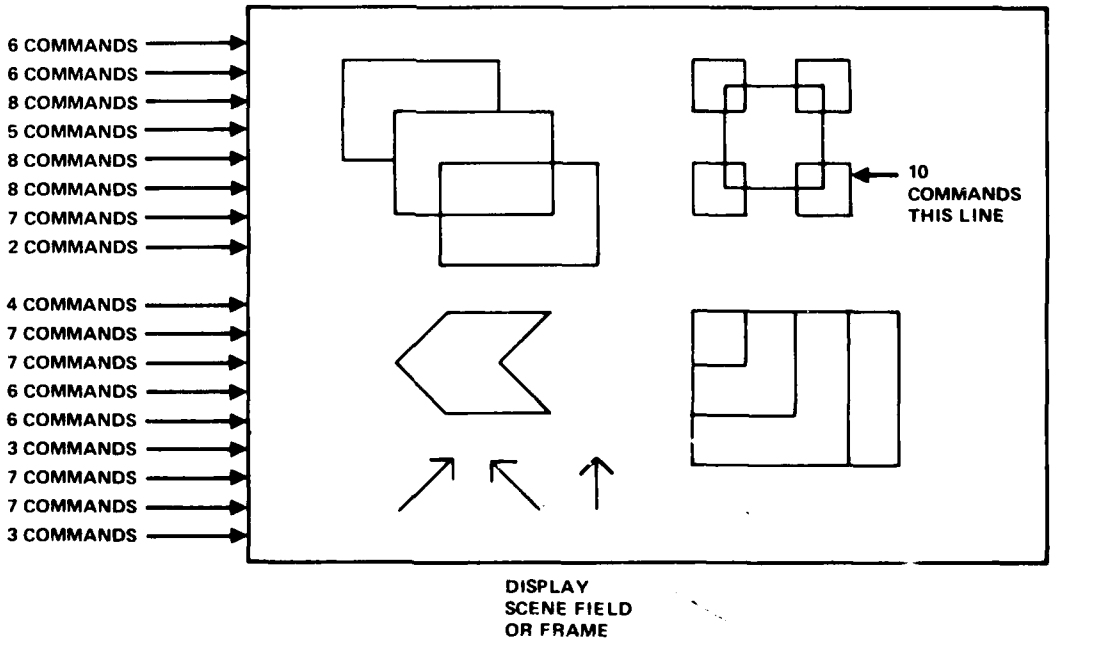
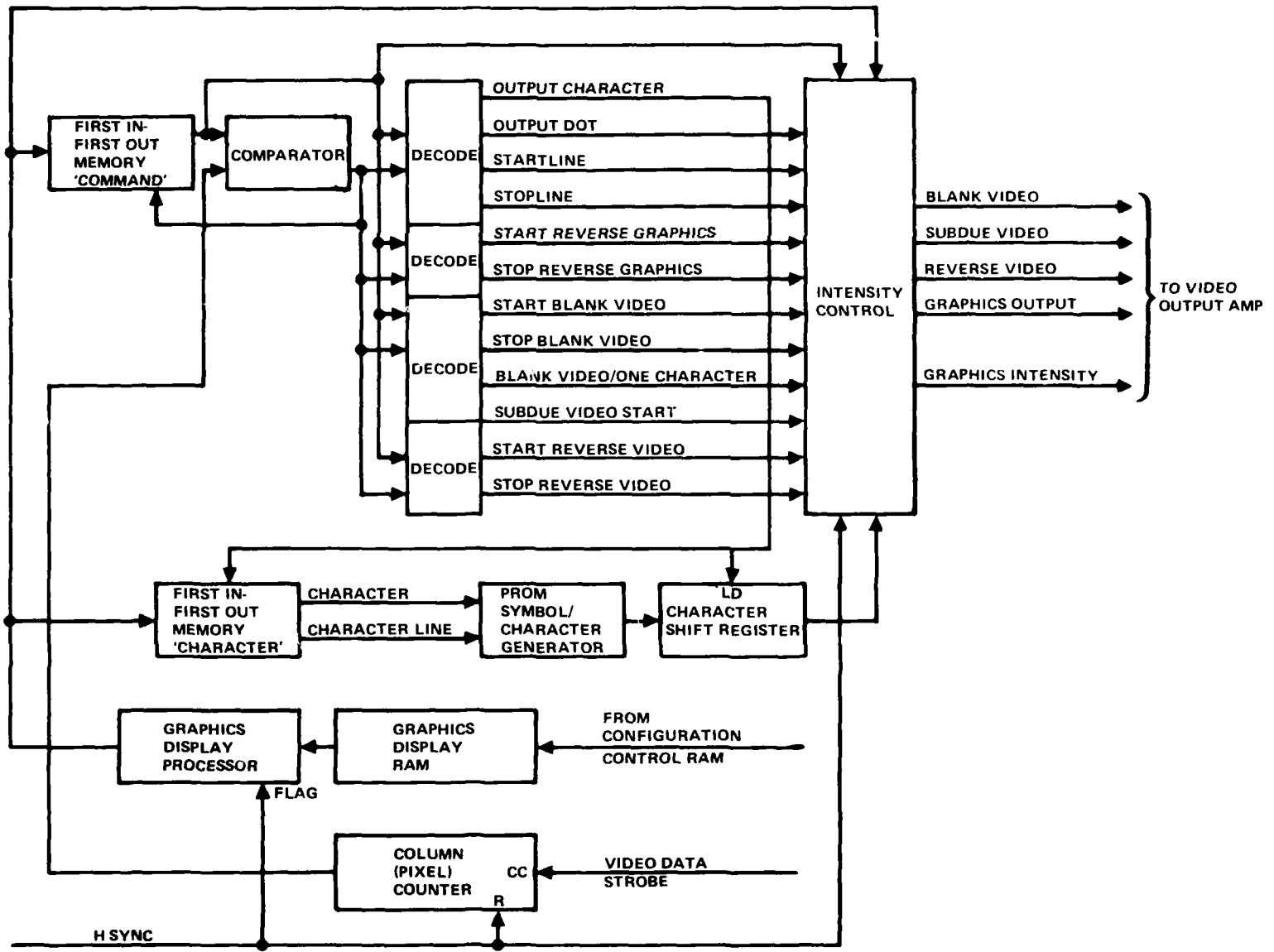


FIGURE 3-30. GRAPHIC DISPLAY PROCESSOR



03-5-23

FIGURE 3-31. GRAPHICS DISPLAY GENERATOR

Processor (GDP) immediately begins to interpret the display requirements into commands for line by line control, by the GDP. These commands are in turn loaded into a 64 word First In First Out (FIFO) memory. Part of the FIFO word is a 10 bit address describing the column (pixel) count at which the command is to be executed. From this it can be seen that commands must be loaded into the FIFO in the sequence in which they are to be executed and that the GDP tests for the FIFO being full, if full no additional commands are sent. FIFO address output compares with the content of the column (pixel) counter the remainder of the FIFO word is decoded into discrete signals for display intensity control. Also, this comparison, signals the next sequential word to be output from the FIFO.

When an output command from the command FIFO, generates the discrete signal "output character" the character shift register is side loaded with an eight bit byte that is the translated value of the character FIFO output. When this shift register is loaded, the next sequential character address output from the character FIFO is made available to the symbol/character generator prom, for translation of the next character to be displayed.

The symbol/character generator PROM is an UV eraseable PROM configured 4,096 words by 8 bits wide. This configuration provides the ability to generate up to 256 different symbols each containing 8 horizontal pixels and up to a maximum of 16 vertical pixels, i.e., up to 128 pixels/symbol. This feature allows for the generation of special symbols, as required, in addition to the full 128 character ASCII set of characters and symbols. The ASCII character set utilizes a 7 x 9 dot matrix display. The symbol/character generator PROM receives its address, for the symbol/character and the display line, of the symbol to be output from the character FIFO memory. The character FIFO in turn has previously been loaded with the eight bit character/symbol identifier and the four bit character line to be displayed. Loading of the character FIFO is accomplished in the same manner as previously described for the command FIFO, from the Graphics Display Processor.

The Intensity Control receives signals from the command Decode, command FIFO, the character shift register and Configuration Control RAM to provide control of the output to the display monitors.

The Graphics Display Controller described above is presently operational in the FACC Video Signal Processing Simulation Laboratory. This concept was chosen for graphics display due to it's flexibility and minimal hardware requirements.

Features of the Intensity Control provides the ability to configure, from the Graphics Display Processor, the default reset (from horizontal sync) that occurs at the start of each new line. For example, if it is required to subdue most of the video data being displayed and only a small portion of the display was to be highlighted, to normal intensity, probably each line would start out with video subdued. In this case a reduced number of commands would be required for a field of data, requiring only two commands per line of the video to be displayed at full intensity and no command for each of the full lines of subdued video. If this provision was not available, one command would be required for the start of each line plus two commands for each line

containing full intensity video. This default configuration applies to video blank, video reverse, graphics reverse and subdue video. Synchronization with horizontal sync to provide reconfiguration of the default reset on a line by line basis is provided. Also, the capability is provided to control the level of intensity to which the video is subdued as well as the brightness of the graphics.

The Graphics Display Controller has been designed to accommodate any aspect ratio (4:3, 2:1 etc.) of display as well as any resolution (875 line, 525 line, etc.) display in either field or interlaced frame mode.

If the Graphics Display Controller is not installed in the system, no adverse effects are caused to the system other than the inability to display graphics information.

The Graphics Display Controller (GDC) is capable of controlling graphics display to a single monitor at a time. Although it is possible to multiplex control to two display monitors, on an alternate field basis, it is not recommended due to the inherent half frequency blink rate. What is recommended is a second GDC for systems that require graphics be displayed on two monitors simultaneously. However, if two display monitors are required to always display identical data with identical timing then it is only necessary to use a single GDC for both displays.

3.2.2.7 Control Panel (B.4). In the AAS system, the control panel will serve as a controller used for initializing, testing, and executing system algorithms. The control panel will consist of a 1553 serial interface, and a control processor with associated memory. A 6802 microprocessor will serve as the central processor. A simplified block diagram of the control panel is given in Figure 3-32. The control panel will be approximately 10 by 8 by 4 inches in physical dimension. The front panel will contain all displays and controls used for system operation. Figure 3-33 illustrates the front panel configuration.

The control panel will have two modes of operation, a modular mode and a system mode. In a system mode of operation, the control panel will supply the system with the necessary information for performing a selected algorithm. The system operation display will indicate the present algorithm being executed. The system operation display will be accessed via the control panel keyboard. Entries by the keyboard will be directed by the system operation controls if a change in operation is desired. In a modular mode of operation, the control panel will be used to perform modular debugging type operations for isolation of a possible problem. The modules which will be accessed in this mode are; system controller, control processor, video preprocessor, and the high speed arithmetic processor. All information will be directed to the monitor display. The keyboard controls will be used for controlling the monitor display. The left row of keyboard controls will control the information which will be supplied by the keyboard. The right row of keyboard controls will indicate the debug operation to be performed. The following table gives a brief description of the control panel switches.

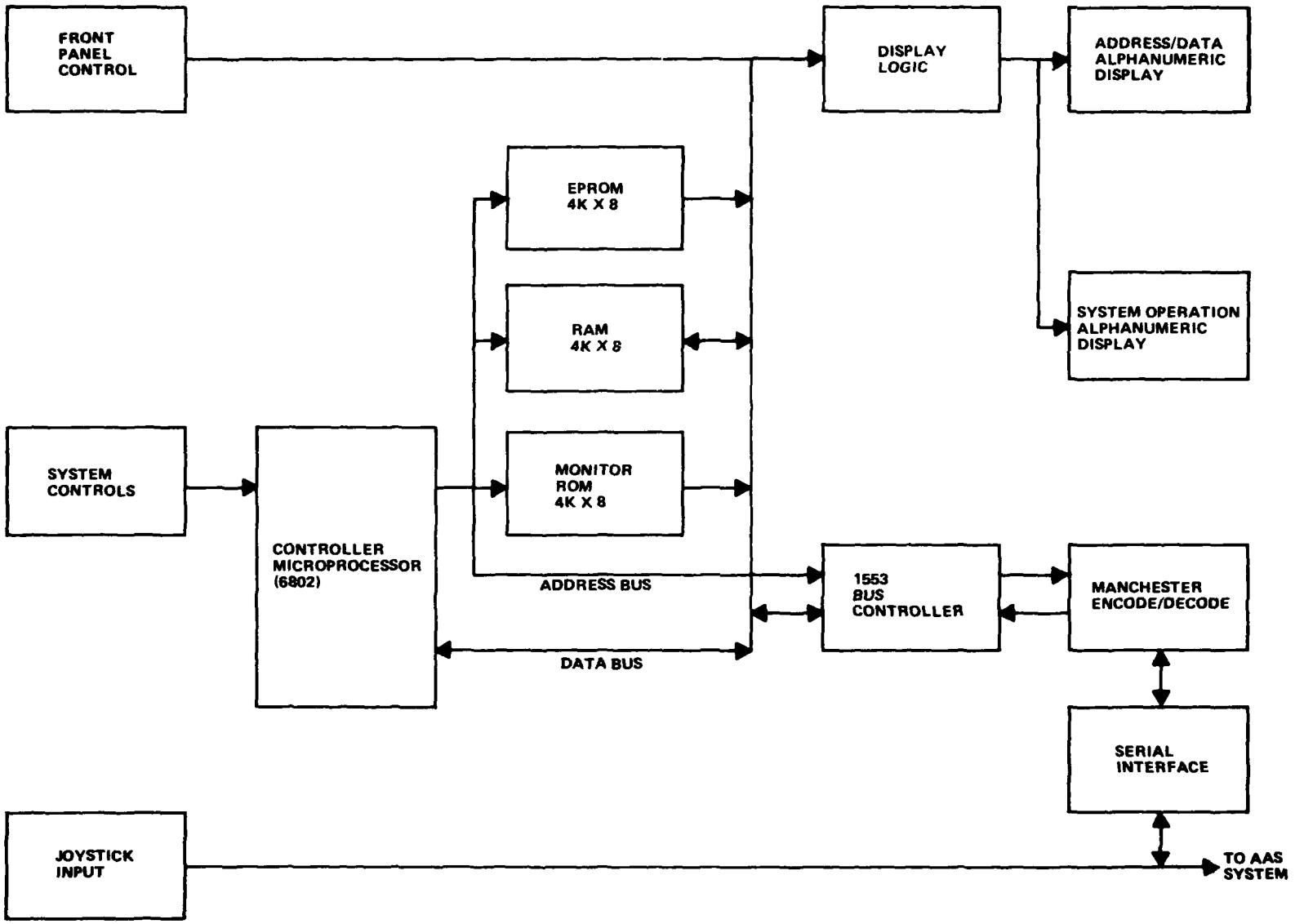


FIGURE 3-32. CONTROL PANEL BLOCK DIAGRAM

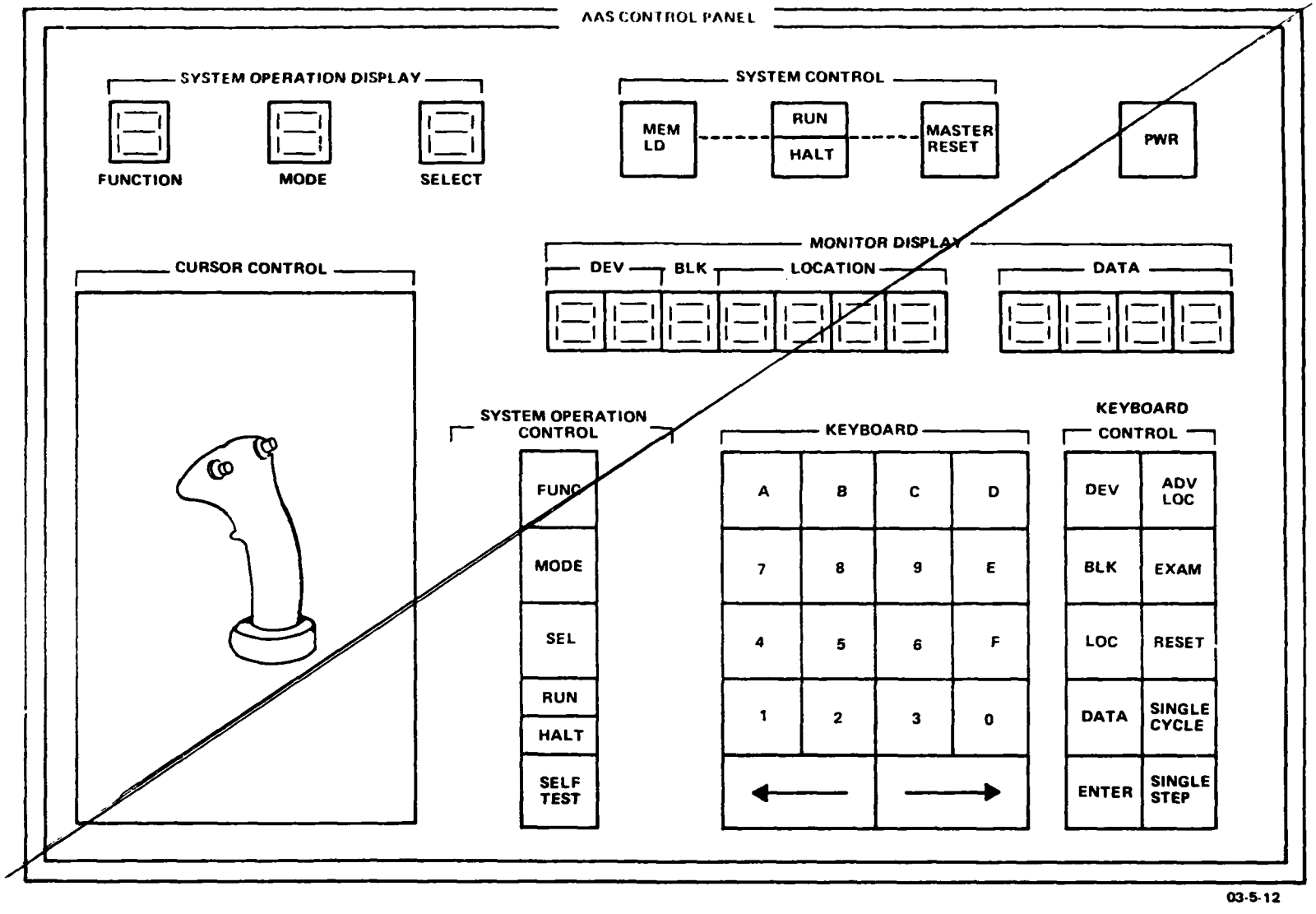


FIGURE 3-33. AAS CONTROL PANEL

RELAYOUT SCHEDULED

Description of Front Panel Switches

HEADING	SWITCH	DESCRIPTION
SYSTEM CONTROL	Master Reset	; initializes system
	Run/Halt	; corresponds to system state
		; end of execution of present function
	MEM LD (Memory Load)	; loads the system with microcode stored in memory module
SYSTEM OPERATION CONTROL	FUNC (Function)	; allows for keyboard entry of function display, and Dev, device, display indicates VP, video processor
	Mode	; allows for keyboard entry of mode display, and indication of DEV display same as above
	SEL (Select)	; allows for keyboard entry of SELECT display
	CP (Control Processor)	; DEV, device, display indicates CP
	IP (Image Processor)	; DEV, device, display indicates IP
	RW/HALT, SELF TEST	; These switches indicate the state of the device displayed in the DEV display
KEYBOARD		; a hexidecimal keyboard with a right and left cursor
KEYBOARD CONTROL	ENTER	; must be pushed before keyboard or system operation display becomes active
	DEV (Device)	; enables the DEV, device, display to be entered from system operation control switches

Description of Front Panel Switches (Contd)

HEADING	SWITCH	DESCRIPTION
	BLK (Block)	; enables the BLK block display to be entered by keyboard
	LOC (Location)	; enables the LOCATION display to be entered by keyboard
	DATA	; enables the DATA display to be entered by keyboard
	EXAM	; used to examine the location specified in Address Display
	ADV LOC (Advance location)	; increments the location of Address Display
	RESET	; reinitialize system without destroying parameters, used main in debugging
	Single Cycle	; a debugging aid which executes a single machine cycle
	Single Step	; a debugging aid which executes a single clock pulse.

3.2.2.8 Special Test Equipment (B.6). A Memory Loader Fixture is provided for loading a transportable memory from a Host Processor computer e.g. the DEC PDP 11/70 or VAX 11/780 fixture communicates as shown in Figure 3-34 with the Host computer over an RS-232 Serial Communications Link as shown in Figure 3-35. The link is a direct cable connection when the Memory Loader Fixture resides local to the Host computer. However, the Memory Loader Fixture may also be located at a remote location by utilization of modems and telecommunications lines. This feature will allow application code for the AAS to be modified in field over standard voice or computer grade telephone lines. The RS-232 link may be selected to run full duplex synchronous or asynchronous at compatible BAUD rates.

The "Memory Loader Fixture" is microprocessor controlled to provide sufficient intelligence to answer the protocol of the host processor as well as providing the timing and control for erasing and programming the transportable memory and communication with the AAS over a serial bus conforming to MIL-STD 1553A. This 1553 Bus provides connection of the Host computer when the system is used in a laboratory environment or at a local or remote ground station.

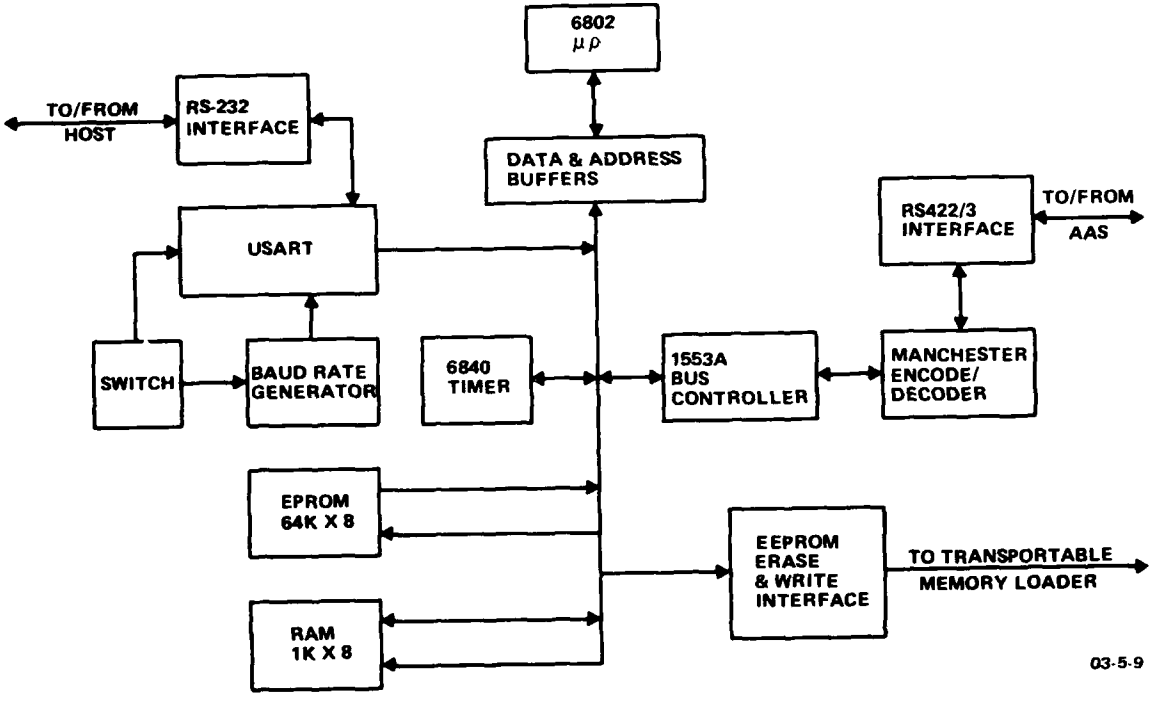


FIGURE 3-34. MEMORY LOADER FIXTURE

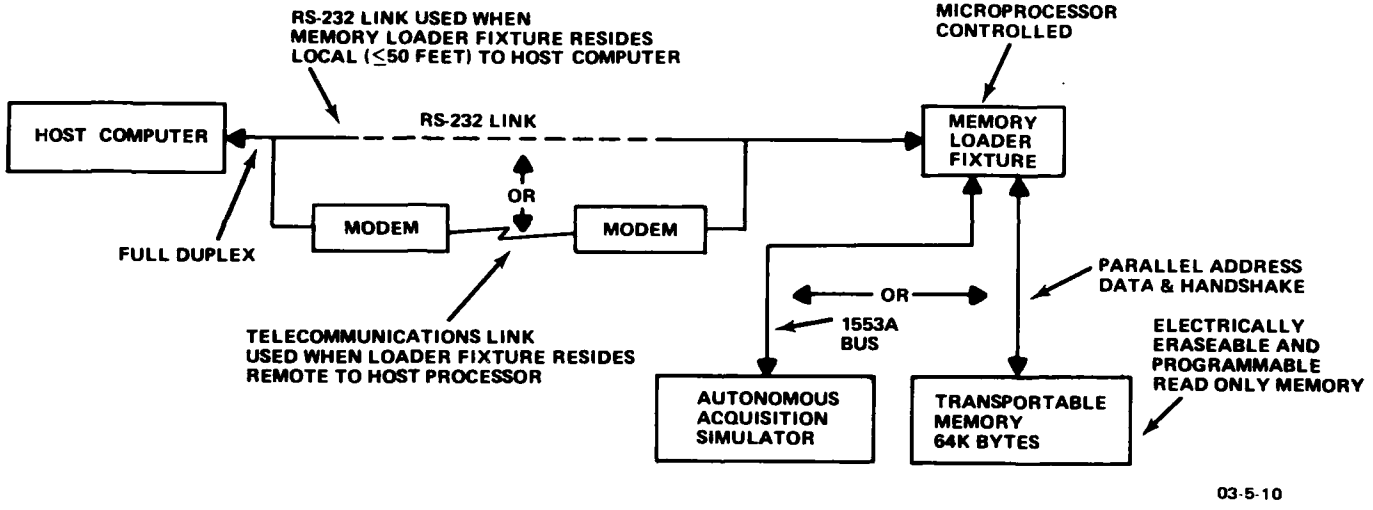


FIGURE 3-35. RS-232 SERIAL COMMUNICATIONS LINK

This provision allows study of algorithm software and verification of systems operation prior to helicopter flight by utilizing the resources of the Host computer.

The Transportable Memory module allows the loading of programs and parameters for flight test of acquisition and track algorithms.

This module consists of a non volitile Bubble memory. The "Transportable Memory" module may be plugged into the Memory Loader Fixture for erasure and programming under the direction of the Host computer via the RS-232 serial bus. The memory module may then be transported and inserted into the AAS which is designed to provide electrical and mechanical connections.

3.2.2.9 Power Requirements and Parts Count (B.1). The power requirements for the baseline AAS design are summarized in Table 3-6. It should be noted that the power listed represents the maximum power required assuming that all the modules are operational. Power saving features have been incorporated in the baseline design to reduce the actual power and correspondingly the cooling requirements. For instance, the baseline design for the Digital Video Preprocessor includes a feature that allows any preprocessor that is not in use to be switched on/off within a field time. Since these modules require a large amount of power, a significant reduction in the actual power required by the AAS is expected.

The parts count for each subsystem for the AAS is summarized in Section 7.3 titled "Support Technical Data" and is not repeated here.

3.2.2.10 Mechanical Design and Packaging (B.1). The AAS processor will be packaged in two boxes. The circuit boards are 10.0 by 15 inches and in one chassis with the multiple output switching power supply in another. Weights are listed in Table 3-7. A remote control panel is cabled into the processor chassis over a

TABLE 3-6. SYSTEM POWER REQUIREMENTS

Voltage Level (Volts)	Max Current (Amp)	Max Power (Watts)
+5.0	90	450
+12.0	20	240
-5.2	46	240
+15	0.5	7.5
-15	0.5	7.5
	Total	945

NOTE:
Presently under reevaluation

TABLE 3-7. WEIGHT BREAKDOWN

Subsystem	Weight, lbs
Chassis and Cables	29.5
Power Supply	16.8
System Controller	3.2
HSDM	13.3
Graphics Display Controller	2.8
DVPP	15.3
Frame Memory	8.3
Control Panel	7.3
Total	96.5

NEW DESIGN FEATURES MAY ALTER THESE VALUES

10-foot cable. The interface is serial thus minimizing cable size. The two boxes will be mounted in a shock mounted rack while in the helicopter test configuration. Refer to Figure 3-36 for a block diagram representation of the chassis, program memory loader, and control panel interconnect. Installation/removal from the helicopter of the AAS system should take less than 5 minutes.

Figure 3-37 gives a detailed pictorial representation of the AAS processor chassis. The box is serviceable from the top with the top also acting as a card holder to minimize board flexing along the 15-inch length. The chassis will hold 25 circuit cards leaving space (5 cards) for expansion. The motherboard will be wire wrapped to allow for flexible interconnects. The motherboard also has rail supports on the bottom to prevent it from flexing during card insertion. For external I/O two 54-pin connectors are mounted on the rear. Each connector has seven possible coaxial inputs for analog I/O. Air circulation is from front to back with six 5-inch fans pulling air through front air filters and over the ICs. To cool the power supply, thus making it more efficient, a fan mounted in the rear of the power supply chassis will also draw air from front to back. The volume of the AAS processor is approximately 5.5 cubic feet.

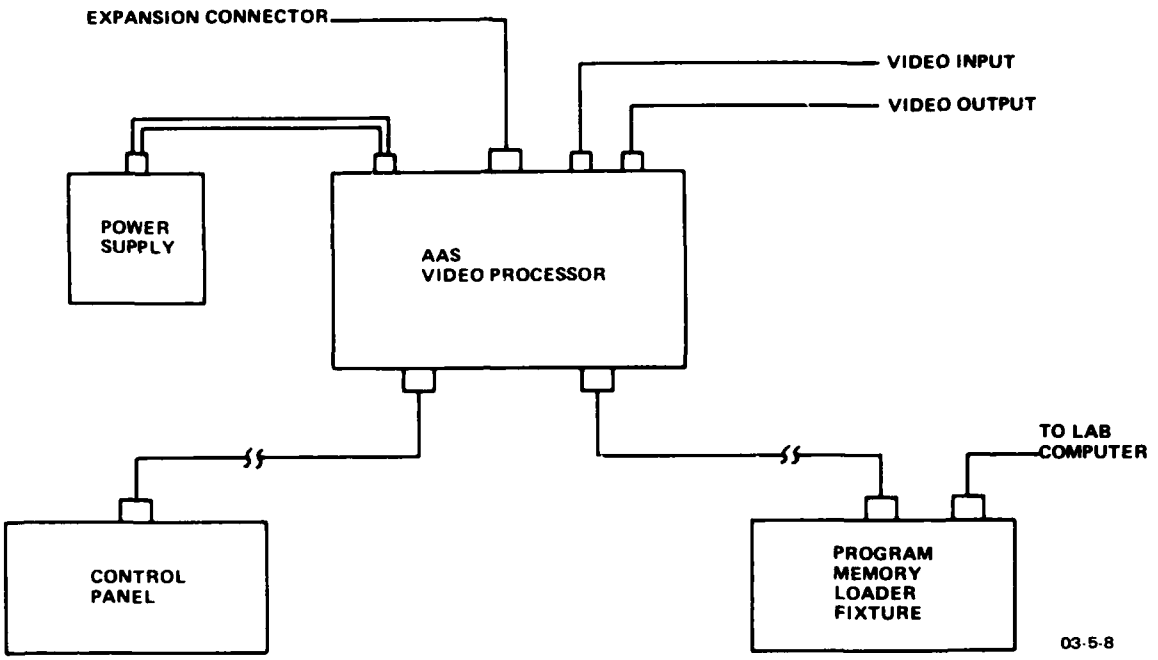


FIGURE 3-36. AAS SYSTEM INTERCONNECT DIAGRAM

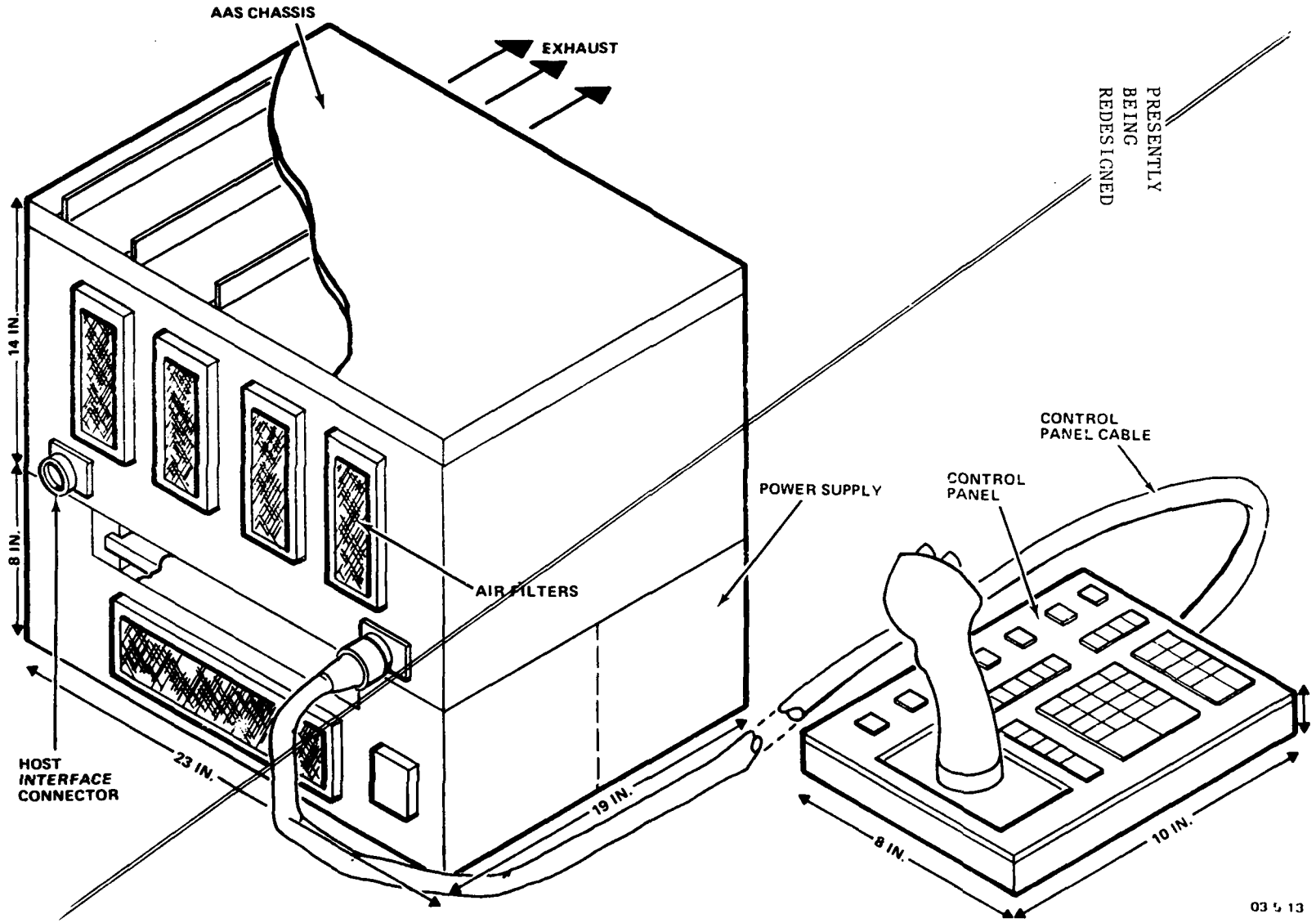


FIGURE 3-37. AUTONOMOUS ACQUISITION SIMULATOR

3.2.3 SYSTEM SOFTWARE DESIGN

The software system proposed for the AAS program closely resembles an existing and on-going software IR&D program currently under development at FACC. That software is being developed at FACC using proven top-down structured programming techniques and a significant portion of it already satisfies many of the requirements for the basic AAS system. Therefore, a considerable amount of the knowledge and experience gained from that IR&D development work will be incorporated into the baseline AAS software design.

The basic software operating system, much of the special support software and some of the diagnostic software will be a direct outgrowth of that IR&D program. The only items of specific software needed for the AAS effort, will be specially coded software routines that implement the unique requirements of the algorithms that will be selected for the system demonstration test.

The primary emphasis in the design of the AAS software system will be placed in four key areas: Speed, Flexibility, Expandability and Maintainability. The system will be fast; it will operate in a near real-time mode. The software system will be extremely flexible. One will be able to program and execute a wide class of algorithms by using simple high level language statements. The system will be expandable to accomodate the inclusion of new and improved algorithms as they are developed by merely adding low-level micro-modules to the existing system micro library. Our basic software design approach utilizes top-down structured programming techniques which ultimately lead to systems that are easy for the user to modify and maintain.

3.2.3.1 Software Design Summary. The software design is intimately connected with the hardware configuration, because the system architecture must be compatible with the simulated algorithms. To ensure the maximum amount of flexibility, while maintaining the high speed necessary for near real-time simulation, we have proposed an ingenious concept that involves three levels of processing for three level programming languages.

The top level is PASCAL, a high level language which allows for a structured programming approach, ease of modification, and a high order language set of supporting system calls. The PASCAL program will run in the DEC LSI 11/23. The second level will be a user defined assembly language, which runs in the bit-sliced control processor. This level allows the software to connect the high order language PASCAL with the microcode in the Pipelined Arithmetic Processor (PAP). The third level is the microcode executed in the PAP which gives us the high speed tight loop arithmetic processing.

The functions performed by each is as follows:

PASCAL (LSI 11/23)

- Define Task Message Lists
- Define User Parameters
- Control Panel Communication

Assembly (Control Processor)

- Check Parameters
- Error Check
- Table Memory Assignments
- Pre-binding Parameters
- Post-binding Parameters
- Micro-Sequencer Start and Definitions
- Bulk Data → Cache Data
- Scratch Pad Parameters
- Post Processing

Micro-Code (PAP)

- Start
- Check Wait/Done
- "DO Loop" Arithmetic Processing

This method of dividing the software tasks between PASCAL, assembly and microcode will give the user maximum flexibility in algorithm development along with the high speed required for realistic simulation.

3.2.3.2 Operating System.

a. Requirements. The software operating system will encompass a system controller and a control processor. The system controller is a DEC LSI 11/23 and the control processor is a custom high speed processor. The system controller will execute a high level PASCAL based language which will form task message packets from a library of algorithm subroutines. These message packets will be passed to the control processor where the control processor will operate on the packets and direct algorithm microsequences to be scheduled and executed in the PAPs. The control processor will perform the processor dependent functions of allocation of resources, processor loading and interrupt servicing.

Multitasking control in the control processor will be required to keep the supporting processors scheduled for execution and to insure that the associated data sets are available when they are needed by the requesting processor. Pre- and post-processing algorithm support in the control processor will perform the necessary address binding and data movement for the algorithm being executed. Message list processing functions will be supported so that

high level language defined algorithm sequences may be stored as lists in the control processor and be subsequently executed on demand by specifying only the list's identifier. This will allow the user to alter algorithm execution parameters easily at the high level language without the need to create new assembly code or microcode.

b. Design. The control processor operating system structure will contain a library of PAP microsequence pointers and their respective pre- and post-processing support code modules for data setup and post-processing of PAP results.

The control processor operating system will be required to execute the task functions associated with individual processors for algorithm queuing and support. Figure 3-38 depicts the basic operating system design structure.

Basic Tasks Descriptions

(1) The system controller communication task will receive and send message packets to setup the control processor and provide input parameter and output display information to the operator through the control panel.

(2) The algorithm list processor task will provide the sequential algorithm sequence processing which has been constructed and commanded by the host processor.

(3) The resource allocator task will perform the housekeeping functions for memory configuration and PAP instruction and cache memory transfers. The resource allocator will also be required to handshake the multitasking transmit/receive dependent algorithm processes in the queue.

(4) The algorithm pre/execute/post processing task will handle the data and microsequence setup to execute a routine in the microsequence library. If the sequence requires execution or post-processing support for PAP data manipulation, this task will provide the support required from the control processor microsequence support library before allowing the next algorithm to be placed into execution.

(5) The video pre-processor support task will be required to setup the memory and pre-processor algorithms on a video field basis. Reprocessing or recursive processing of the same or different fields in frame memory will also be supported by this task. Symbol display (gates, crosshairs, and text) will be setup, processed and directed to the video pre-processor at the vertical blanking time.

(6) The error log task will be required to save the pertinent information about the processors and the algorithms being processed when a processing error occurs. The task will either continue processing or abort depending upon the severity of the error and an indication of the error will be sent to the host processor for operator display or debugging if necessary.

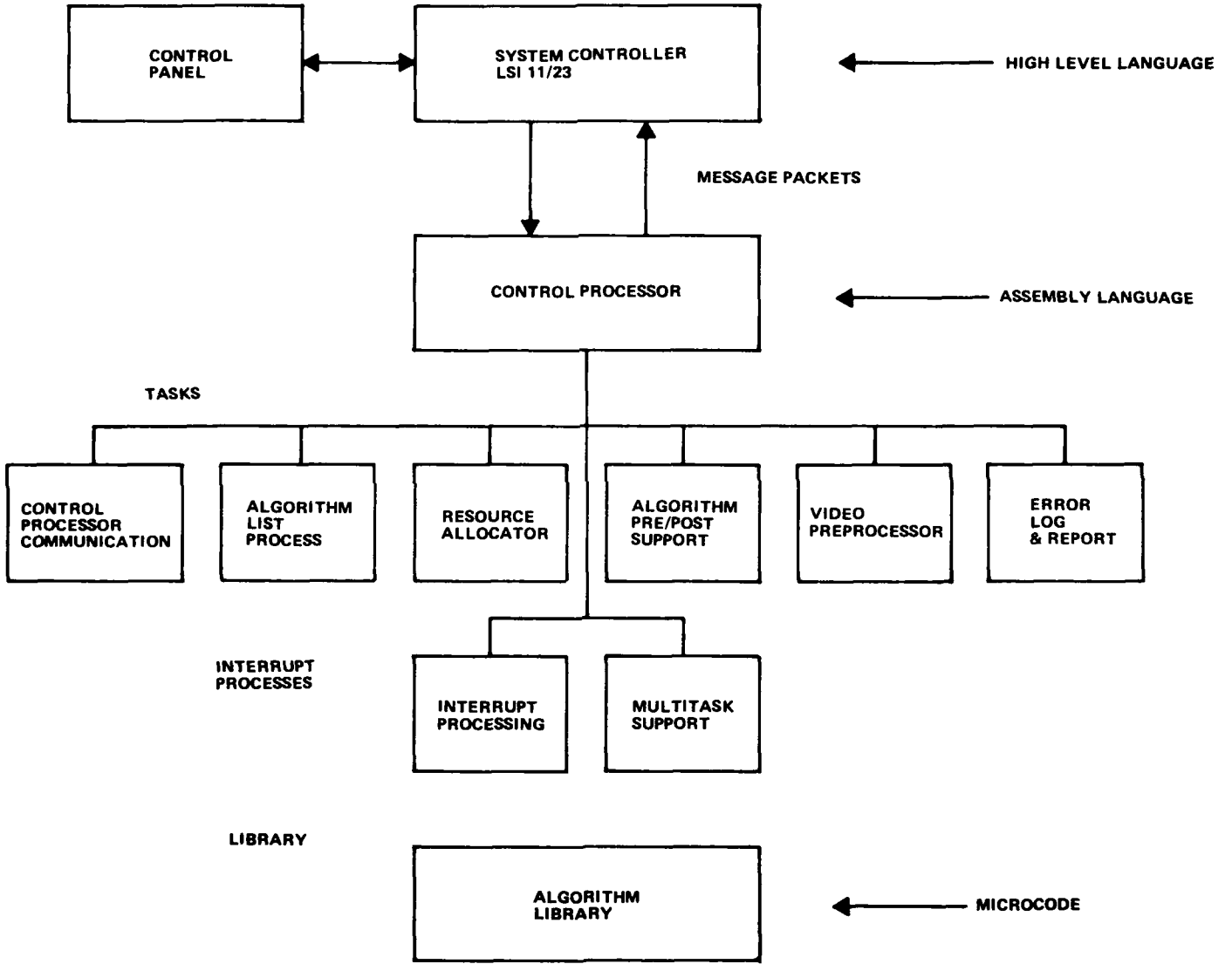


FIGURE 3-38. AAS OPERATING SYSTEM DESIGN STRUCTURE

3.2.3.3 High Level Image Processing Language (B.6). Since the AAS system is used for algorithmic simulation, it is imperative that the system be capable of accepting a high level language and translating to microcode those statements which fully exploit the hardware features provided for image processing.

High level language capability has the advantages of easily transforming algorithms into execution code. In addition, high level language can be modified and maintained easier. Also developing and debugging programs in a high level language is much easier and less expensive for the user.

The present trend in programming methodology is toward structured programming and this is primarily due to the following benefits gained by using this type of approach:

(1) Emphasizes hierarchical approach which leads to easier program management

(2) Improves program readability and maintainability

(3) Produces code with fewer errors

(4) Reduces documentation time because the programs themselves will be self-documenting

(5) Increases programmer productivity

In the proposed effort, the structured approach will be emphasized and to support this, the PASCAL language is offered on the AAS system. PASCAL has high level control structures that help to keep the flow within each module linear, and hence, easily traceable.

PASCAL will be used to specify computation and image processing tasks to the image processing hardware as follows. The image processing complex has a repertoire of macros which can be invoked by using procedure calls within the PASCAL program. Thus, if it is required to perform a histogram, the program segment to perform this may appear as follows:

```
      .
      .
      .
HISTOGRAM (SA, NB, L1, L2)
      .
      .
      .
PROCEDURE HISTOGRAM (SA, NB, L1, L2)
BEGIN
      .
      .
WRITE (IPF, M#, SA, NB, L1, L2)
      .
      .
END
```

The procedure HISTOGRAM outputs to the image processing file (IPF) the following information: M#, the macro number which identifies the histogram task in the image processing hardware; SA, the starting address of the image; NB, number of bins for the histogram; and L1, L2, grey levels that specify the region of interest.

Whenever IPF receives data through the WRITE statements, the parameters are transferred to the hardware. We can also monitor the progress and extract variables from the hardware using statements like:

READ (IPF, OUT1, OUT2,...)

Each macro has a list of input and output variables and when a statement of the above form is encountered, values for OUT1, OUT2,... are read from the IPF. However, when the macro is initiated by the procedure call, the latter has to specify the output variables it requires.

The above example describes the essence of our approach in translating PASCAL high level code to microcode of the image processing hardware. As can be seen, with this approach, one has the capability to specify in PASCAL through procedure calls complex tasks that are to be performed in the hardware. During the contract phase, procedures that implement a representative set of the functions and algorithms listed in Table I of the RFQ will be developed using specially designed macros. Additionally, attentions will be paid to procedures that are useful for debugging and monitoring. These procedures will include display of memory maps in the image processing hardware.

3.2.3.4 Programming Support Software.

a. Microprogram Assembler. The microprogram assembler that has been selected for the AAS program will be the TRANSPORTABLE META assembly (TMA) developed and distributed by Step Engineering. Some of the features that led to the selection are:

- Compatible with AMDASM
- Two versions: Intel MDS
  ANSI standard Fortran IV
- Available on CDC Timesharing Service
- Complete cross reference tables
- Easy to use error listings
- Fully supported
- PROM or MICROWORD TM output
- Direct interface to Step-2

TMA is primarily a development aid for designers working on microprogrammed systems. It permits the user to generate a unique symbolic language which can then be utilized for program assembly. The TMA is compatible with AMD's AMDASM program, but comes in two versions. The first is Intel development system comparible while the second will run on a 16-bit (or larger) minicomputer with a Fortran Compiler, e.g., DEC PDP 11 and Data General Nova series computers.

The Meta Assembly Software Package consists of three separate programs as follows: Definition Program, Assembly Program, and Output Formatter Program.

Definition Program: This program allows the user to define instruction mnemonics and their associated formats. Instruction lengths may vary from 1 to 128 bits. Symbolic Constants and reserved words may also be defined in the Definition Program. An instruction format is defined by breaking the microword into fields and defining the fields as constants, don't care bits, or variables which are filled in at assembly time. Default values and certain permanent attributes may also be assigned to variable fields at Definition time. The Definition Program produces an output listing and a disk file consisting of the defined symbols and instruction mnemonics. This Definition file is used by the Assembly program as a reference when assembling a program.

Assembly Program: This program operates like a traditional assembler. A symbolic source program utilizing the mnemonics and symbols defined in the Definition Program is read as input, and a listing and object module are generated as output. The Assembler provides symbolic addressing, relative addressing, paged addressing, and other features found in typical assembly programs. The instruction syntax and assembler directives are compatible with those utilized by AMD in its literature and software products. Additional directives have been implemented to provide for versatile listing and output controls.

Conditional assembly statements are provided in both the Definition and Assembly programs. These statements may be nested up to 16 levels and can be made dependent on general expression. A full cross reference table is also provided in both programs.

Some features of the TMA are particularly helpful when assembling code for microprogrammable machines. The existence of don't care bits and instruction overlaying are included among these features. Bits of a microword which are not relevant to a particular instruction format may be defined as don't care bits. Don't care bits are printed as X's on the listing and do not have to be defined until the Output Formatter program is executed. An instruction format with don't care bits can be overlayed with other instruction formats. Therefore when useful, an instruction format can be used to define only part of the microword, padding out the word with don't care bits.

Output Formatter Program: The program reads the object module file produced by the Assembly program and translates the format into one that can be read by a PROM programmer or STEP-2. STEP's Microword^{TM} format BPNF and ASCII hexadecimal PROM format are supported. Users may specify PROMs of any width and

length within the object module as well as the value of don't care bits. Any or all PROMs may be listed and/or punched. To output information to STEP-2 Firmware Integration and Test Station (FITS™) the object module is listed in Microword™ format over an RS232 link.

A convenient feature of the TMA is the ease of error correction. Each line in the listing is assigned a reference line number which is used for editing purposes. The error listing also refers to the line number, and output directives permit error messages to be listed following the line referred to, thus permitting rapid correction of source code.

The Fortran version of the program is available as a card set (approx 6000 cards), as a card image on magnetic tape, or as a diskette for LSI-11 systems.

It requires a computer with:

(1) A Fortran VI compiler

(2) A word length of at least 16 bits

(3) 32k words of Random Access Memory (in most systems these programs can be run in an overlayed mode if the required memory is not available)

The Intel development system version comes on either a single or double density diskette and requires a development system with LSI 11 and 64K Bytes ROM.

The AAS program will make use of the version on diskette for the LSI-11 system. In addition, code would be developed on FACC in-house Data General System and VAX 11/780 computer system using the TMA.

b. Macro Instruction Library. The AAS system shall provide the user with the capability of controlling and specifying image processing tasks to the hardware complex through the Host Computer. The Host Computer shall be programmed in a high level language (PASCAL) that has the ability to define image processing tasks by macro instructions. A limited macro library shall be provided to make the system operational. The user can change or expand the macro library according to his needs. The macro library shall consist of the following four categories:

- Supervisory Macros. These macros allow the Control Processor to assign tasks to the Video Preprocessor and High Speed Microprocessors. They enable the Control Processor to communicate information to the System Controller.

- Data Transfer Macros. These macros are used to control the data transfers within the system such as transfer of images within the frame memory or transfer of operands to the Control Processor.

- General Function Macros. These macros are user oriented and since the proposed simulator is mainly used in the context of image processing, vector and matrix operations are good candidates for implementation. A typical (but not necessarily complete) list of this type of macros is the following.

    . VADD (vector add)

    . VMUL (vector multiply)

    . INPROD (inner product of two vectors)

    . OUTPROD (outer product of two vectors)

    . MATADD (matrix add)

    . MATMUL (matrix multiply)

    . SMULV (scaler multiply of vector)

    . VROT (rotate vector)

    . VSHIFT (shift vector)

- Algorithm Execution Macros. These macros are primarily application oriented and perform simple to complex algorithms or tasks exploiting fully the hardware features (such as parallelism and pipelining) and available memory. A typical list is given in the following.

    . DEFIMAG (defines images and subimages)

    . CORREL (correlates two images)

    . HISTO (finds a histogram of an image)

    . CONVMASK (defines convolution masks for performing matched and double gate filtering, edge detection and image enhancement)

    . MOMFEAT (evaluates moment features for classification)

    . BINARIZE (binarizes an image for a given threshold)

    . MEDFILT (performs a median filter on the image)

    . ROTIMAG (rotates the image)

The number and type of macros to be supplied with the AAS system will be more completely defined during the development phase.

The syntax and number of macros to be supplied with the AAS system will be identified and defined during the development phase on FACC's on-going internal IR&D program. The illustration of a simple macro follows. This outline is for a vector dot product.

(1) The System Controller will invoke a data transfer macro to load cache 0 and cache 1 of the pipelined Arithmetic Processor (PAP) with the two vectors.

(2) The Host Processor shall invoke a dot product macro, such as DOTAB (C0, C1, N).

(a) DOTAB will identify a microcoded routine that performs the dot product.

$$A \cdot B = A_1 B_1 + A_2 B_2 + \text{----} + A_N B_N$$

(b) C0 and C1 will identify to the microcoded routine that the vectors are stored in PAP caches 0 and 1.

(c) The N will identify the number of elements in each vector.

(3) The Control Processor will respond to the invoked macro by initializing the PAP counter one to the value of N, and then execute the DOTAB microcode routine.

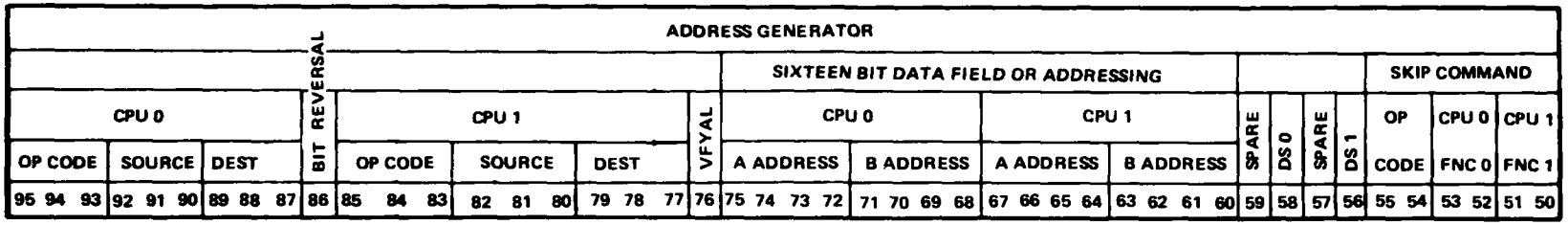
(4) The microcoded routine will leave the resultant product in the scratch Pad.

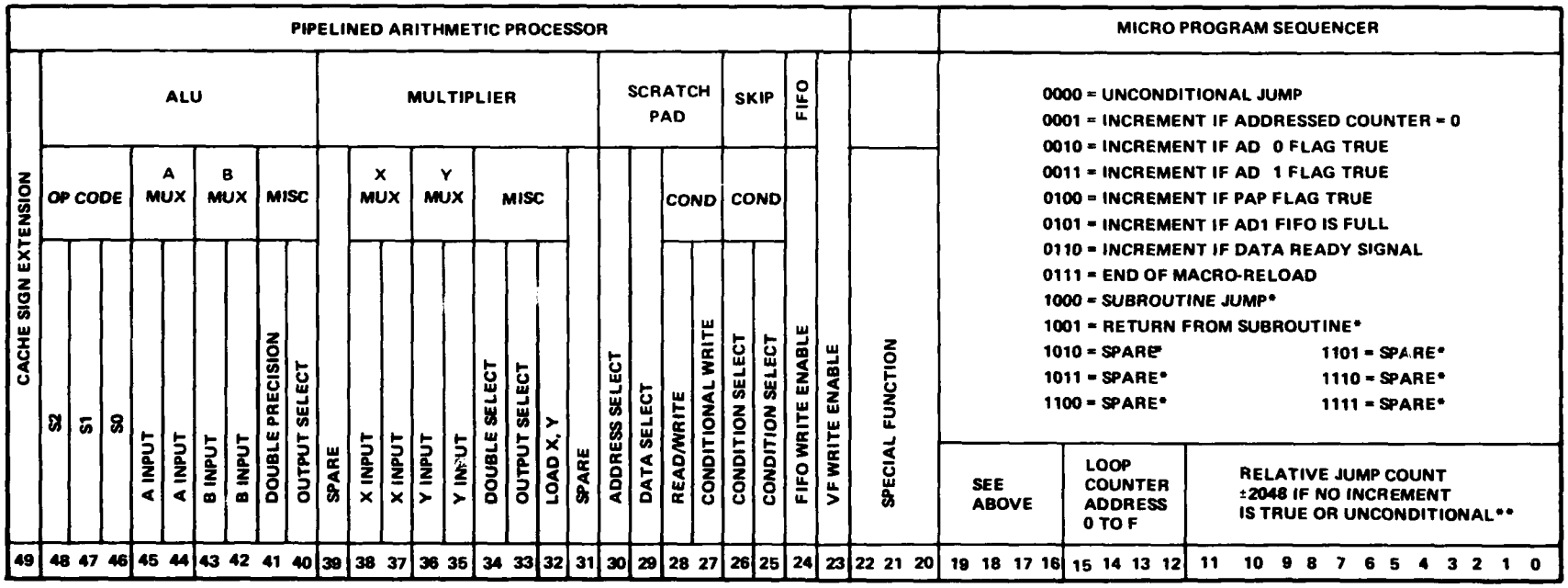
An illustration of the micro instruction word and the microcoded example routine DOTAB is provided in Figure 3-39.

c. Diagnostic Software. The AAS system will provide for three levels of diagnostic testing. The lowest level of testing shall utilize the Built-in Test Equipment (BITE) feature of the AAS system. The next level of testing shall be a functional subunit test. The highest level of test shall be a system go/no go system test. Upon operator activation of a system test key on the control panel, the AAS system shall automatically perform all three levels of testing.

The first test performed by the AAS system software shall be the activation of the BITE feature in the following modules:

- Micro Sequencer
- Address Generator
- Bulk Memory
- Pipelined Arithmetic Processor





* FUTURE GROWTH OPTION, NOT AVAILABLE

** FUTURE GROWTH OPTION, RELATIVE JUMP COUNT IS ±127 LOCATIONS

FIGURE 3-39. INSTRUCTION WORD FORMAT

```
LINE     ADDR   MVIPS ASSEMBLER DOT PRODUCT                              PAGE     1

DATE    1-13-81
TIME   15:13:38

    1   00000   TITLE                    MVIPS ASSEMBLER DOT PRODUCT
    2   00000   ;
    3   00000   ;*****************************************************************
    4   00000   ;  DOTPD SR
    5   00000   ;
    6   00000   ;
    7   00000   ;                  FORD AEROSPACE AND COMMUNICATIONS CORPORATION
    8   00000   ;                          AERONUTRONICS DIVISION
    9   00000   ;                             MVIPS PROJECT
   10   00000   ;
   11   00000   ;
   12   00000   ;           S = (A0 * B0) + (A1 * B1) + (A2 * B2)
   13   00000   ;
   14   00000   ;  THIS ROUTINE CALCULATES THE DOT PRODUCT OF TWO
   15   00000   ;  VECTORS CONTAINING THREE ELEMENTS EACH AND STORES THE
   16   00000   ;  RESULT IN THE SCRATCH PAD (SP).  IT ASSUMES CACHE 0
   17   00000   ;  CONTAINS VECTOR A AND CACHE 1 CONTAINS VECTOR B IN
   18   00000   ;  THE FOLLOWING ORDER
   19   00000   ;
   20   00000   ;           CH0:              CH1:
   21   00000   ;           0:  A0            0:  B0
   22   00000   ;           1:  A1            1:  B1
   23   00000   ;           2:  A2            2:  B2
   24   00000   ;
   25   00000   ;  AG0 IS USED AS A POINTER INTO THE CACHES AND AG1 IS USED
   26   00000   ;  AS A POINTER INTO THE SP.  AG1 IS SET TO 1 THRU OUT THE
   27   00000   ;  ENTIRE ROUTINE.
   28   00000   ;    THE ROUTINE IS SET UP FOR AN INFINITE LOOP
   29   00000   ;
   30   00000   ;
   31   00000   ;  MODULE ID:  4.3.2.2
   32   00000   ;
   33   00000   ;       M. L. TUMBLESON
   34   00000   ;       04-22-80
   35   00000   ;  MODIFIED BY:  M. L. TUMBLESON          1-13-81
   36   00000   ;
   37   00000   ;*****************************************************************
   38   00000   ;
   39   00000   LIST      B,X
   40   00000   NOLIST L
   41   00000   ;
   42   00000   ;
   43   00000   ;
   44   00000   ;THIS IS THE BEGINING OF THE LOOP
   45   00000   ;     CACHE POINTER AG0 SET TO ZERO FOR
   46   00000   ;     THE FIRST ELEMENT PAIR (A0,B0)
   47   00000   ;
   48   00000   A0          ADD     DZ,Q                ;AG0   0 --> AG0
   49   00000   /&          HLD1                        ;AG1:  HOLD
   50   00000   /&          RADRS   4H#0,4H#0,4H#0,4H#0    ;DATA IS 0
   51   00000   /&          NVFYL                       ;NO VFM WRITE
   52   00000   /&          NCRE                        ;PAP:  NO CARRY EXT.
   53   00000   /&          HLDA                               HOLD ACCUM
   54   00000   /&          HLDM                               HOLD MULT
```

NOTE: THIS IS THE LATEST LISTING

```
DATE  [illegible]
TIME  [illegible]

00000  /&         RD                      ;        HOLD SP
00000  /&         JMP                     ;MS  INCREMENT
00001
00001  ;
00001  ;[illegible] ON ONE [illegible] E [illegible] AGO LATCH
00001  ;
00001  A1         HC                      ;NOP
00001  /&         HL[illegible]
[illegible]       H[illegible]            ;NO VFM WRITE
[illegible]
00001             H[illegible]
[illegible]       HL
[illegible]       R[illegible]
00001  /&
00002  ;
00002  ;
00002  ;CACHE POINTER AG [illegible] R NEXT PAIR [illegible]
00002  ;INITIALIZE [illegible] POINTER AG1 TO 1 FOR SUM
00002  ;MULTIPLY (A0 * B[illegible]
00002  ;
00002  A3         AD[illegible] [illegible]   ;AG0  A0 + 1 -> A00
00002  /&         AL[illegible]           ;AG1  1 -> AD1
00002  /&         RA[illegible] [illegible] 4H#[illegible] 4H#1   ;DATA IS 1
00002  /&         NV[illegible]           ;NO VFM WRITE
00002  /&         N[illegible]            ;PAP  NO CARRY EXT
00002  /&         HLDA                    ;     HOLD ACCUMULATOR
00002  /&         L[illegible] [illegible] IY   ;  B * A -> M
00002  /&         R[illegible]            ;     HOLD SP
00002  /&         JM[illegible]           ;MS   INCREMENT
00003  ;
00003  ;
00003  ;[illegible] P [illegible] PRODUCT
00003  ;
00003  A5         HL[illegible]           ;AG0  HOLD
00003  /&         HL[illegible]1          ;AG1  HOLD
00003  /&         NV[illegible]L          ;NO VFM WRITE
00003  /&         NCRE                    ;PAP  NO CARRY EXT
00003  /&         HLDA                    ;     HOLD ACCUMULATOR
00003  /&         HLOM     LSP            ;     SELECT LSP
00003  /&         RDSP                    ;     HOLD SP
00003  /&         JMP                     ;MS   INCREMENT
00004  ;
00004  ;
00004  ;CACHE POINTER AG0=2 FOR LAST PAIR (A2,B2)
00004  ;SET SUM = A0 * B0 AND MULTIPLY (A1 * B1)
00004  ;STORE FIRST RUNNING SUM IN SP(1)
00004  ;
00004  A4:        ADC0     7Q [illegible]  ;AG0  AD0 + 1 -> AD0
00004  /&         HLD1                    ;AG1  HOLD
00004  /&         NVFYL                   ;NO VFM WRITE
00004  /&         NCRE                    ;PAP  NO CARRY EXT
00004  /&         HLDA                    ;     HOLD ACCUMULATOR
00004  /&         LDXY     [illegible] CIY   ;  B * A -> M
00004  /&         WRSP     ADSP,MLSP      ;     M -> SP(1)
```

```
  109   00004  /&        JMP                              ;MS: INCREMENT
  110   00005  ;
  111   00005  ;
  112   00005  ;READ FIRST SUM SP(1) INTO SP LATCH
  113   00005  ;
  114   00005  A5        HLD0                             ;AG0: HOLD
  115   00005  /&        HLD1                             ;AG1: HOLD
  116   00005  /&          NVFYL                          ;NO VFM WRITE
  117   00005  /&        NCRE                             ;PAP: NO CARRY EXT.
  118   00005  /&        HLDA                             ;     HOLD
  119   00005  /&        HLDM      LSP                    ;       HOLD MULT. & SELECT LSP
  120   00005  /&        RDSP        ;HLDSP               ;       READ SP INTO LATCH
  121   00005  /&        JMP                              ;MS: INCREMENT
  122   00006  ;
  123   00006  ;
  124   00006  ;SUM = SUM + (A1 * B1) AND MULTIPLY (A2 * B2)
  125   00006  ;
  126   00006  A6        HLD0                             ;AG0: HOLD
  127   00006  /&          HLD1                           ;AG1: HOLD
  128   00006  /&          NVFYL                          ;NO VFM WRITE
  129   00006  /&          NCRE                           ;PAP: NO CARRY EXT.
  130   00006  /&          ADD       MA,SPB               ;     M + SP(1) -> AC
  131   00006  /&        LDXY        C0, C1Y              ;     B2 * A2 -> M
  132   00006  /&          RDSP                           ;     HOLD SP
  133   00006  /&          JMP                            ;MS: INCREMENT
  134   00007  ;
  135   00007  ;
  136   00007  ;STORE SECOND RUNNING SUM IN SP(1)
  137   00007  ;
  138   00007  A7          HLD0                           ;AG0: HOLD
  139   00007  /&          HLD1                           ;AG1: HOLD
  140   00007  /&          NVFYL                          ;NO VFM WRITE
  141   00007  /&          NCRE                           ;PAP: NO CARRY EXT.
  142   00007  /&          HLDA                           ;     HOLD ACCUM.
  143   00007  /&          HLDM                           ;     HOLD MULT
  144   00007  /&          WRSP      ;WRSP                ;     AC -> SP(1)
  145   00007  /&          JMP                            ;MS: INCREMENT
  146   00008  ;
  147   00008  ;
  148   00008  ;SELECT LSP PORTION OF PRODUCT & READ SECOND SUM
  149   00008  ;SP(1) INTO LATCH
  150   00008  ;
  151   00008  A8          HLD0                           ;AG0: HOLD
  152   00008  /&          HLD1                           ;AG1: HOLD
  153   00008  /&          NVFYL                          ;NO VFM WRITE
  154   00008  /&          NCRE                           ;PAP: NO CARRY EXT.
  155   00008  /&          HLDA                           ;     HOLD ACCUM.
  156   00008  /&          HLDM      LSP                  ;     SELECT LSP
  157   00008  /&          RDSP        ;HLDSP             ;     READ SP INTO LATCH
  158   00008  /&          JMP                            ;MS: INCREMENT
  159   00009  ;
  160   00009  ;
  161   00009  ;SUM = SUM + (A2 * B2
  162   00009  ;
```

```
LINE     ADDR    MVIPS ASSEMBLER DOT PRODUCT                              PAGE

DATE     1-13-83
TIME    15 13 30

 163    00009   A9.         HLDO                        ;AGO: HOLD
 164    00009   /&          HLD1                        ;AG1: HOLD
 165    00009   /&          NVFYL                       ;NO VFM WRITE
 166    00009   /&          NCRE                        ;PAP: NO CARRY EXT.
 167    00009   /&          ADD     MA  MB              ;     M + SP(1) -> AC
 168    00009   /&          HLDM                        ;     HOLD MULT.
 169    00009   /&          RDSP                        ;     HOLD SP
 170    00009   /&          JMP                         ;MS:  INCREMENT
 171    0000A   ;
 172    0000A   ;
 173    0000A   ;STORE FINAL RUNNING SUM IN SP(1)
 174    0000A   ;
 175    0000A   AJO         HLDO                        ;AGO: HOLD
 176    0000A   /&          HLD1                        ;AG1: HOLD
 177    0000A   /&          NVFYL                       ;NO VFM WRITE
 178    0000A   /&          NCRE                        ;PAP: NO CARRY EXT
 179    0000A   /&          HLDA                        ;     HOLD ACCUM.
 180    0000A   /&          HLDM                        ;     HOLD MULT.
 181    0000A   /&          WRSP    ADS+                ;     AC -> SP(1)
 182    0000A   /&          JMP                         ;MS:  INCREMENT
 183    0000B   ;
 184    0000B   ;
 185    0000B   ;GO BACK TO BEGINING OF LOOP
 186    0000B   ;
 187    0000B   AJJ         HLDO                        ;AGO: HOLD
 188    0000B   /&          HLD1                        ;AG1: HOLD
 189    0000B   /&          NVFYL                       ;NO VFM WRITE
 190    0000B   /&          NCRE                        ;PAP: NO CARRY EXT.
 191    0000B   /&          HLDA                        ;     HOLD ACCUM.
 192    0000B   /&          HLDM                        ;     HOLD MULT.
 193    0000B   /&          RDSP                        ;     HOLD SP
 194    0000B   /&          JMP     12H#FF6             ;MS:  JUMP BACK 11
 195    0000C   ;                                       ;     INSTRUCTIONS
 196    0000C   END

  TOTAL ASSEMBLY ERRORS =     0
```

OBJECT MODE

```
00000  0001111110X101100  1111XXXX  0000000  XXXXXXXXXXXXX10  00000001X0000010
00000  X0100111OXXX0000  00000000  0000001
00001  1011001111X101100  1111XXXX  XXXXXXX  XXXXXXXXXXXXX10  00000001X0000010
00001  X0100111OXXX0000  00000000  0000001
00002  1110101110X000111  1101000X  0000000  XXXXXXXXXXXXX10  00000001X0000011
00002  X0100111OXXX0000  00000000  0000001
00003  1011001111X101100  1111XXXX  XXXXXXX  XXXXXXXXXXXXX10  00000001X0000000
00003  X0100111OXXX0000  00000000  0000001
00004  1110101110X101100  1111XXXX  XXXXXXX  XXXXXXXXXXXXX10  00000001X0000011
00004  X0011111OXXX0000  00000000  0000001
00005  1011001111X101100  1111XXXX  XXXXXXX  XXXXXXXXXXXXX10  00000001X0000000
00005  X0101111OXXX0000  00000000  0000001
00006  1011001111X101100  1111XXXX  XXXXXXX  XXXXXXXXXXXXX10  11010001X0000011
00006  X0100111OXXX0000  00000000  0000001
00007  1011001111X101100  1111XXXX  XXXXXXX  XXXXXXXXXXXXX10  00000001X0000010
00007  X0111111OXXX0000  00000000  0000001
00008  1011001111X101100  1111XXXX  XXXXXXX  XXXXXXXXXXXXX10  00000001X0000000
00008  X0101111OXXX0000  00000000  0000001
00009  1011001111X101100  1111XXXX  XXXXXXX  XXXXXXXXXXXXX10  11010001X0000010
00009  X0100111OXXX0600  00000000  0000001
0000A  1011001111X101100  1111XXXX  XXXXXXX  XXXXXXXXXXXXX10  00000001X0000010
0000A  X0111111OXXX0000  00000000  0000001
0000B  1011001111X101100  1111XXXX  XXXXXXX  XXXXXXXXXXXXX10  00000001X0000010
0000B  X0100111OXXX0000  00001111  1110110
```

HEX LISTING

```
1F2C  F000  0002  0102  270C  0001
B3AC  F000  0002  0102  2700  0001
EB0C  0000  1002  0103  270C  0001
B2AC  F000  0002  0100  270C  0001
xB2C  F000  C00x  0103  1F0C  0001
B3AC  F000  0002  0100  2FCC  0001
B3AC  F000  0002  D103  270C  0001
B3AC  F000  0002  0102  3FCC  0001
B3AC  F000  0002  0100  2FCC  0001
B3AC  F000  0002  D102  270C  0001
B3AC  F000  0002  0102  3FCC  0001
B3AC  F000  0002  0102  270C  0FFC
```

Upon completion of the BITE, the diagnostic software shall report any failures found by the test equipment to the operator.

The next test performed by the AAS system software shall be the functional subunit test. The functional subunit test shall consist of individual self tests for the following subunits:

- Control Panel (6802) CPU test
- Program Transport Module (6802) CPU test
- System Interface (6802) CPU test
- 1553 Data Bus Data Transfer Verification
- System Controller Self Test
- Control Processor Self Test
- Microsequencer Self Test
- Address Generator Self Test
- Pipelined Arithmetic Processor Self Test
- Bulk Memory Self Test
- Video Preprocessor Self Test

The diagnostic self tests shall report a failure of its respective subunit for operator notification.

The final test performed by the diagnostic software shall be a system level scenario test. This test shall consist of processing a predetermined scene, and comparing selected processing outputs with expected outputs. Any discrepancy between the processed and the expected outputs resulting from the scene shall be treated as a system no-go condition. The amount of processing performed shall be minimal and intended to exercise all system units in an operational manner.

3.3 INTEGRATED MEMORY PROCESSOR (Option)

3.3.1 INTRODUCTION AND SUMMARY

The Integrated Memory Processor (IMP) developed for the Autonomous Acquisition Simulator will utilize the design architecture of existing hardware developed specifically for digital image processing applications. Its design is based on using dedicated hardware to implement general convolutional algorithms needed in image processing applications. Its application will enhance the processing capabilities of the AAS System by expanding it to include applications of near realtime image rotations, translations, minification, magnification, and variable user define filters. The following information presents the design characteristics of the IMP and describes areas which will require upgrading for the AAS application.

3.3.2 FUNCTIONAL DESCRIPTION

The functional capability of the Integrated Memory Processor developed for the AAS System will include the current image processing capabilities of performing functions ranging from image rotation, scaling, translations, minification, magnification, and warping using either Bilinear or Cubic Spline interpolation. It will also include the current capability to performing user defined convolutional filters ranging from a 2x2 matrix up to a 15x15 matrix. All of which are performed under near realtime conditions utilizing direct memory access of display frame buffers. These capabilities currently exist for applications to displays of 512 by 512 pixels but will be modified to accommodate expansions for applications of 1024 by 1024 pixels. Figure 3-40 illustrates the functional components of the Integrated Memory Processor hardware.

3.3.3 SYSTEM CONFIGURATION

The IMP will be configured within the realtime environment of the AAS data flow and will require access to two frame buffer memories when employed. Figure 3-41 illustrates how the IMP will be implemented for the AAS System.

The design of the IMP will be divided into five modular components. Each will perform a major function of the IMP's efforts. This division will give the implementation of the IMP a unique modular structure which will allow simplification of failure analysis and system expansion.

3.3.4 PERFORMANCE

The IMP for the AAS System will be designed from the current near realtime image processing application. This system is currently capable of performing functions of rotation, scaling, translation, magnification, minification and non-recursive filters well within specifications required for AAS. The following paragraph defines the performance characteristics currently available and the expansion required for the AAS System.

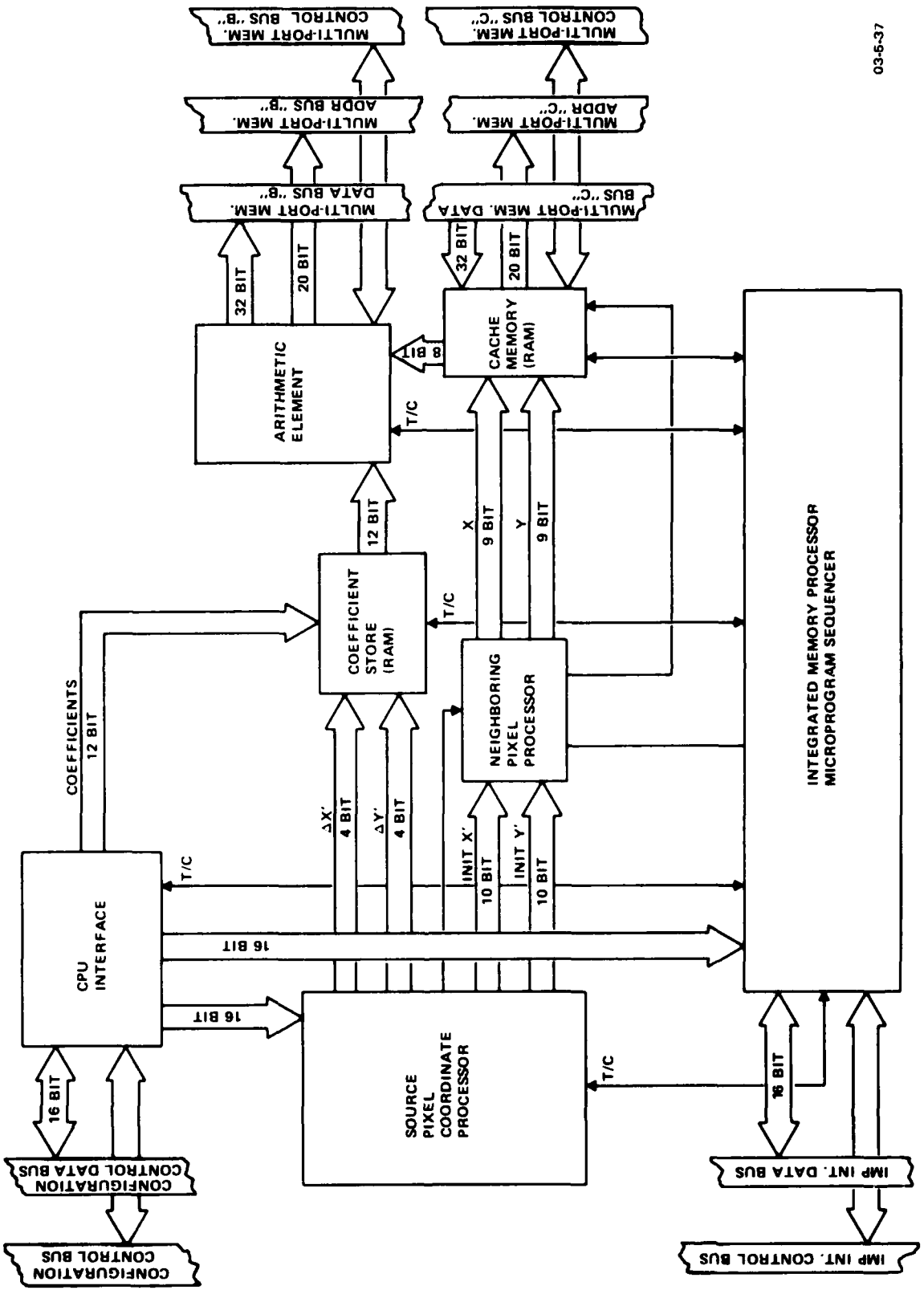


FIGURE 3-40. INTEGRATED MEMORY PROCESSOR FUNCTIONAL BLOCK DIAGRAM

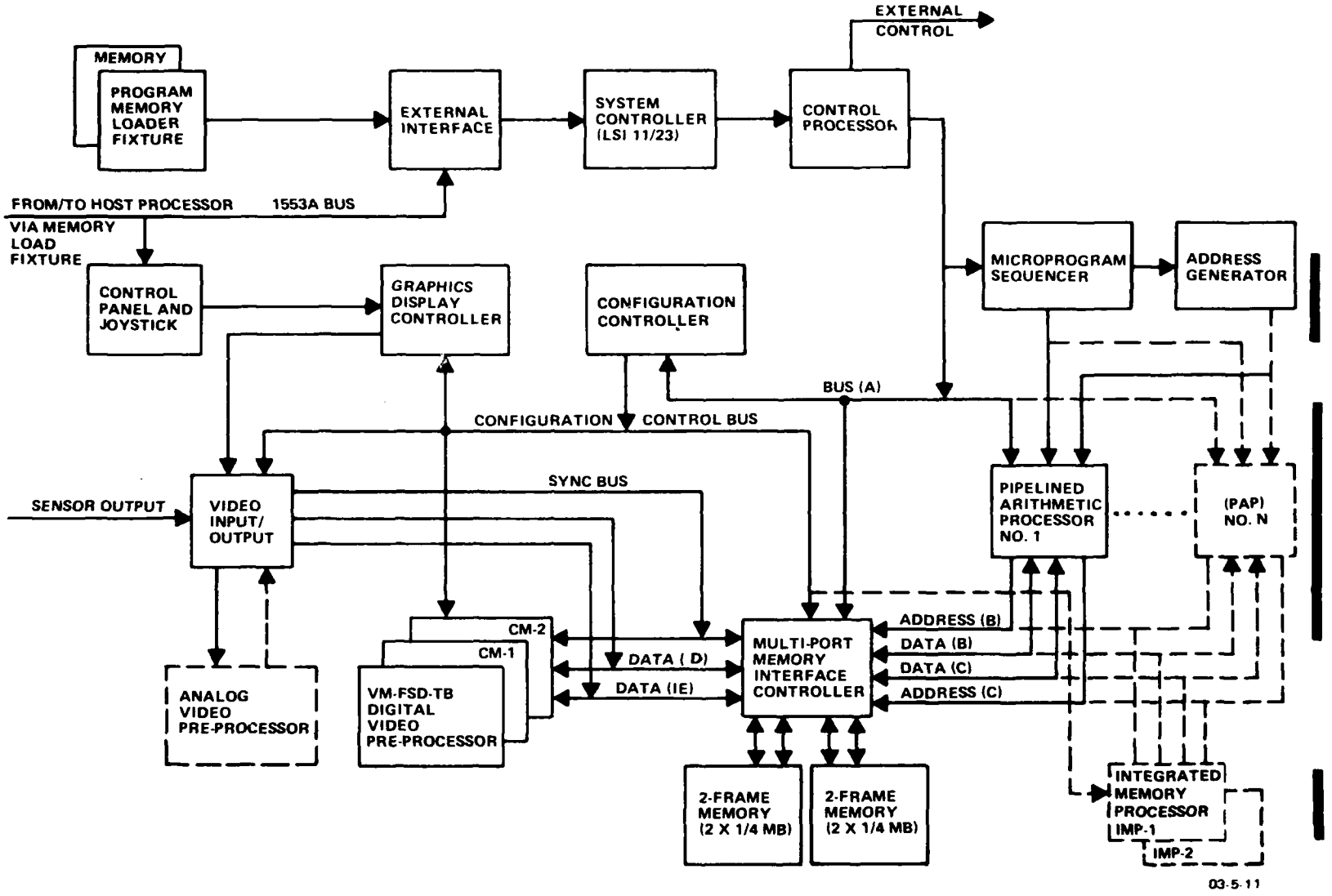


FIGURE 3-41. INTEGRATED MEMORY PROCESSOR AAS SYSTEM ENVIRONMENT

3.3.4.1 Functional Applications. The AAS IMP will employ existing hardware architecture to implement the following functions:

FUNCTION	CURRENT CAPABILITY	AAS EXPANSION
Rotation	0 to $\pm$ 180° in 1024 increments	Same
Translation	0 to $\pm$ 512 pixels in increments of 1/16 pixel	Expand to 1024 Pixels
Minification/ Magnification	1/16 to 16 in increments of 1/16	Same
Interpolation	Bilinear and Cubic Spline	Same
Special Transformations	Warping	Same
Filter	Non-Recursive Convolutional (2x2 to 15x15)	Non-Recursive Convolutional (2x2 to 64x64)

Bilinear or Cubic Spline interpolation will be applicable to any first order functions (i.e., rotation, translation, magnification and minification) as specified by the AAS System Controller.

3.3.4.2 Hardware. The hardware implementation of the Integrated Memory Processor will incorporate existing operational hardware which utilizes the most advanced state-of-the-art LSI and MSI digital integrated circuitry. Schotty TTL bipolar logic will be used in most situations. Principal multiplying functions will be implemented with high speed LSI multipliers MPY-16HJ and MPY-12HJ types available from TRW with off-the-shelf availability of 150 nanoseconds and 110 nanoseconds respectively. It's process control logic will be implemented with the most recent control oriented processing architecture using 2900 family logic to provide the most flexible integration of hardware and software possible.

3.3.4.3 Software Requirements. The IMP will be provided complete with operational firmware but will require functional software interfacing. This will, however, be limited to functional commands to the IMP, coefficient data required for implementing filters, and first or second order processor coefficients.

3.3.5 INTERFACES

The IMP will require interface communication links to the configuration Controller and to 2 frame buffer memories via the Multiport Memory Interface Control.

The interface to the configuration Controller will consist of communication of all functional commands, first and second order coefficients, configuration commands, filter dimensions, and filter coefficients.

The Multiport Memory Interface Controller interface will be responsible for direct memory access to display data for the IMP. This interface will consist of two (2) 20 bit address links in which the IMP will pass address information to the designated working store frame buffer over one bus, and response addresses over the other. It will also consist of two (2) 32 bit data interface links one for accessing display data from the working store buffer and one to pass response data to the response buffer.

Each access to the working store buffer will request 8 consecutive pixels each. These 8 pixels are considered a display vector and are stored within the Active Pixel Memory of the IMP Cache. This also compliments the IMPs Cache application by minimizing external request which slow down the total IMP processing cycle.

3.3.6 PHYSICAL CHARACTERISTICS

The Integrated Memory Processor will require three AAS System Chassis card slots. The distribution of power for each functional unit is estimated below:

	FUNCTIONAL UNIT	POWER * PER UNIT
(1)	CPU Interface/Processor Control	20 Watts
(2)	Source Pixel Coordinate Processor	40 Watts
(3)	Neighboring Pixel Coordinate Processor and Coefficient Table Look-Up	30 Watts
(4)	Cache	40 Watts
(5)	Arithmetic Element	20 Watts
	TOTAL	150 Watts

*All estimates are based of Vcc = +5.0 Vdc.


DATE
FILMED
8